

Sujet n° 1

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
2. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. a pour valeur FAUX
 - c. entraîne l'affichage d'un message d'erreur
 - d. n'a pas de valeur
3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(&a, &b);`
 - b. `echange(a, b);`
 - c. `echange(a++, b++);`
 - d. `echange(*a, *b);`
4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void exchange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void exchange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void exchange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void exchange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
5. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère `'A'`
6. Les instructions

```
i=0;
while(i<10)
    printf("%i ", i);
    i++;
```

 - a. vont afficher 9 nombres
 - b. vont afficher 11 nombres
 - c. vont afficher 10 nombres
 - d. vont boucler indéfiniment

7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 45
9. `if` est :
 - a. Un identificateur du langage C
 - b. Un mot-clef du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un opérateur du langage C
10. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - c. Augmente la valeur de `a` quelque soit `a`
 - d. Affiche bonjour quelque soit `a`
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 62
 - c. 68
 - d. 54
12. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand `a` est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier pair
13. `printf("%c", 'A')`
 - a. Affiche le caractère `'A'`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
14. `if (a%2 == 0) printf("bonjour") ;`
 - a. N'affiche rien (quelque soit la valeur de `a`)
 - b. Affiche bonjour quand `a` est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand `a` est un entier pair
15. Après `char c ; c='a' ; c=c+1 ;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut `'A'`
 - c. `c` vaut `'b'`
 - d. `c` vaut `'a'`

16. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
17. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
18. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - le contenu de la variable
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
 - 111011010100000
 - 111011010101000
 - 111011010101111
20. L'expression `a<=1`
- Réalise une affectation
 - Diminue de 1 la valeur de *a*
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`

Sujet n° 2

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100000
- c. 111011010100110
- d. 111011010101111

2. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Réalise une affectation
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Diminue de 1 la valeur de a

3. Après `char c ; c='a' ; c=c+1 ;`

- a. c vaut `'b'`
- b. c vaut `'A'`
- c. c vaut `'a'`
- d. Un message d'erreur s'affiche

4. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur FAUX
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur VRAI

5. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère `'A'`

6. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour quelque soit a
- b. Affiche bonjour et augmente la valeur de a quelque soit a
- c. Augmente la valeur de a quelque soit a
- d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a

7. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand a est un entier pair
- b. N'affiche rien (quelque soit la valeur de a)
- c. Affiche bonjour quand a est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
10. Les instructions `i=0 ;`
- ```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- vont afficher 10 nombres
  - vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment
11. `if` est :
- Un opérateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
  - `echange(*a, *b) ;`
  - `echange(&a, &b) ;`
  - `echange(a++, b++) ;`
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
14. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
15. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`

16. `printf("%c", A)`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
17. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. ce vers quoi pointe la variable
  - d. le contenu de la variable
18. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de *a* tout simplement
  - b. l'adresse de la variable *a*
  - c. ce vers quoi pointe le pointeur *a*
  - d. n'a pas de sens
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 68
  - b. 54
  - c. 58
  - d. 62
20. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de *a*)
  - c. Affiche bonjour quand *a* est un entier impair
  - d. Affiche bonjour quand *a* est un entier pair

# Sujet n° 3

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100110
- c. 111011010101000
- d. 111011010100000

3. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 100
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 10

4. `if` est :

- a. Un mot-clef du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un opérateur du langage C
- d. Un identificateur du langage C

5. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

6. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 54
- b. 58
- c. 68
- d. 62

7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. l'adresse de la variable `a`
- c. ce vers quoi pointe le pointeur `a`
- d. n'a pas de sens

8. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
9. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
  - l'adresse d'un pointeur sur la variable
  - le contenu de la variable
  - ce vers quoi pointe la variable
10. Les instructions `i=0;`
- ```

while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - a pour valeur FAUX
 - n'a pas de valeur
 - a pour valeur VRAI
12. Après `char c; c='a'; c=c+1;`
- c vaut 'a'
 - c vaut 'b'
 - c vaut 'A'
 - Un message d'erreur s'affiche
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables a et b on doit écrire :
- `echange(*a, *b);`
 - `echange(&a, &b);`
 - `echange(a++, b++);`
 - `echange(a, b);`
14. `printf("%c", A)`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
15. L'expression `a<=1`
- Réalise une affectation
 - Diminue de 1 la valeur de a
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. la valeur de `a` tout simplement
 - c. l'adresse de la variable `a`
 - d. n'a pas de sens
17. `printf("%i", A)`
- a. Affiche le caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
18. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Affiche bonjour quand `a` est un entier impair
 - d. Affiche bonjour quand `a` est un entier pair
19. `if (a%2 = 0) printf("bonjour");`
- a. Affiche bonjour quand `a` est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier pair
20. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère `'A'`

Sujet n° 4

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut `'A'`
 - c. `c` vaut `'b'`
 - d. `c` vaut `'a'`
2. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand `a` est un entier pair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier impair
3. `if (a%2 = 0) printf("bonjour") ;`
 - a. N'affiche rien (quelque soit la valeur de `a`)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand `a` est un entier impair
 - d. Affiche bonjour quand `a` est un entier pair
4. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
5. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. ce vers quoi pointe la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
6. L'expression `a<=1`
 - a. Réalise une affectation
 - b. `A` pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Utilise les opérateurs `<` et `=`
 - d. Diminue de 1 la valeur de `a`
7. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère `'A'`

8. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Augmente la valeur de a quelque soit a
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b);`
 - `echange(a++, b++);`
 - `echange(&a, &b);`
 - `echange(a, b);`
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 54
 - 68
 - 58
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - n'a pas de valeur
 - a pour valeur FAUX
12. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
13. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
14. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont afficher 9 nombres
15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. ce vers quoi pointe le pointeur `a`
- c. l'adresse de la variable `a`
- d. la valeur de `a` tout simplement

18. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

19. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :

- a. `111011010101111`
- b. `111011010100000`
- c. `111011010101000`
- d. `111011010100110`

20. `if` est :

- a. Un identificateur du langage C
- b. Un mot-clef du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un opérateur du langage C

Sujet n° 5

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
 - a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. l'adresse d'un pointeur sur la variable
 - c. ce vers quoi pointe la variable
 - d. le contenu de la variable
2. `printf("%c", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
3. `if` est :
 - a. Un identificateur du langage C
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
4. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

 - a. vont afficher 10 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 11 nombres
 - d. vont afficher 9 nombres
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. la valeur de `a` tout simplement
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(&a, &b) ;`
 - b. `echange(*a, *b) ;`
 - c. `echange(a, b) ;`
 - d. `echange(a++, b++) ;`

7. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010100110
 - 111011010100000
 - 111011010101111
8. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 68
 - 62
 - 54
10. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand a est un entier pair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
11. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
12. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
13. Après `char c ; c='a' ; c=c+1 ;`
- c vaut 'b'
 - Un message d'erreur s'affiche
 - c vaut 'a'
 - c vaut 'A'
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur FAUX

15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

16. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de a)
- b. Affiche bonjour quand a est un entier impair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand a est un entier pair

17. L'expression `a--`

- a. Diminue de 1 la valeur de a
- b. Utilise les opérateurs `<` et `=`
- c. Réalise une affectation
- d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon

18. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 100

19. On suppose que `a` a été déclarée par `int *a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. l'adresse de la variable `a`
- c. n'a pas de sens
- d. la valeur de `a` tout simplement

20. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

Sujet n° 6

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de a
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010100110
 - c. 111011010101000
 - d. 111011010101111
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
4. `printf("%i", A)`
 - a. Affiche le caractère '`A`'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère '`A`'
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 62
 - c. 68
 - d. 54
6. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 100
7. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. le contenu de la variable
 - d. l'adresse d'un pointeur sur la variable

8. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un opérateur du langage C
 - Un identificateur du langage C
9. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont afficher 9 nombres
10. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
11. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
13. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour quelque soit *a*
14. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
15. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - la valeur de *a* tout simplement
 - ce vers quoi pointe le pointeur *a*
 - l'adresse de la variable *a*

16. `printf("%c", A)`
- a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
17. Après `char c ; c='a' ; c=c+1 ;`
- a. *c* vaut 'A'
 - b. *c* vaut 'a'
 - c. Un message d'erreur s'affiche
 - d. *c* vaut 'b'
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur FAUX
19. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand *a* est un entier impair
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- a. `echange(a, b) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(*a, *b) ;`

Sujet n° 7

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
2. `if` est :
 - a. Un opérateur du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un mot-clef du langage C
 - d. Un identificateur du langage C
3. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

 - a. vont afficher 9 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 10 nombres
 - d. vont afficher 11 nombres
4. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
5. `if (a<5) printf("Bonjour"); a=a+1 ;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - b. Affiche bonjour quelque soit `a`
 - c. Augmente la valeur de `a` quelque soit `a`
 - d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(*a, *b) ;`
 - b. `echange(a, b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(&a, &b) ;`

7. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
8. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
10. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
11. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier pair
 - Affiche bonjour quand *a* est un entier impair
12. `if (a%2 = 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 58
 - 68
 - 54
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - a pour valeur FAUX
 - n'a pas de valeur

15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

16. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- c. Diminue de 1 la valeur de a
- d. Réalise une affectation

17. Après `char c; c='a'; c=c+1;`

- a. c vaut `'b'`
- b. c vaut `'A'`
- c. Un message d'erreur s'affiche
- d. c vaut `'a'`

18. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable A

19. `printf("%i", A)`

- a. Affiche le code ASCII du caractère `'A'`
- b. Affiche le caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le code ASCII du caractère stocké dans la variable A

20. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100000
- c. 111011010101111
- d. 111011010100110

Sujet n° 8

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
2. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère 'A'
3. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand a est un entier pair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier impair
4. L'adresse d'une variable c'est :
 - a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. ce vers quoi pointe la variable
 - c. le contenu de la variable
 - d. l'adresse d'un pointeur sur la variable
5. `if` est :
 - a. Un identificateur du langage C
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
6. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier impair
 - c. Affiche bonjour quand a est un entier pair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`

7. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont afficher 11 nombres
b. vont afficher 10 nombres
c. vont boucler indéfiniment
d. vont afficher 9 nombres
8. L'expression `a<=1`
a. Réalise une affectation
b. Diminue de 1 la valeur de a
c. Utilise les opérateurs `<` et `=`
d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
9. `printf("%c", A)`
a. Affiche le caractère dont le code ASCII est stocké dans la variable A
b. Affiche le code ASCII du caractère stocké dans la variable A
c. Affiche le code ASCII du caractère `'A'`
d. Affiche le caractère `'A'`
10. L'expression `(0 == 7%3) || (1 == 9%3)`
a. a pour valeur FAUX
b. n'a pas de valeur
c. entraîne l'affichage d'un message d'erreur
d. a pour valeur VRAI
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
a. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
c. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
12. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
a. Permettent de d'affecter à `t[10]` la valeur 10
b. Permettent de d'affecter à `t[10]` la valeur 45
c. Permettent de d'affecter à `t[10]` la valeur 0
d. Permettent de d'affecter à `t[10]` la valeur 100
13. Après `char c ; c='a' ; c=c+1 ;`
a. Un message d'erreur s'affiche
b. c vaut `'A'`
c. c vaut `'b'`
d. c vaut `'a'`
14. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
a. 111011010101000
b. 111011010100110
c. 111011010101111
d. 111011010100000

15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(&a, &b) ;`
- d. `echange(*a, *b) ;`

16. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 54
- b. 58
- c. 62
- d. 68

17. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère `'A'`

18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. n'a pas de sens
- c. l'adresse de la variable `a`
- d. ce vers quoi pointe le pointeur `a`

19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable `a`
- b. la valeur de `a` tout simplement
- c. ce vers quoi pointe le pointeur `a`
- d. n'a pas de sens

20. `printf("%i", A)`

- a. Affiche le code ASCII du caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

Sujet n° 9

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont boucler indéfiniment
 - b. vont afficher 11 nombres
 - c. vont afficher 10 nombres
 - d. vont afficher 9 nombres
2. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 0
3. `if` est :
 - a. Un identificateur du langage C
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
4. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. a pour valeur FAUX
 - c. n'a pas de valeur
 - d. entraîne l'affichage d'un message d'erreur
5. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
6. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - c. Augmente la valeur de `a` quelque soit `a`
 - d. Affiche bonjour quelque soit `a`

7. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
 - 58
 - 54
 - 62
8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - la valeur de `a` tout simplement
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
 - `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(a, b) ;`
10. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le contenu de la variable
11. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier pair
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
13. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
14. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de `a`

15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

16. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable *A*
- b. Affiche le caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

17. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable *A*
- c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- d. Affiche le caractère 'A'

18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100000
- c. 111011010100110
- d. 111011010101000

19. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le code ASCII du caractère stocké dans la variable *A*
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le caractère 'A'

20. Après `char c; c='a'; c=c+1;`

- a. *c* vaut 'b'
- b. Un message d'erreur s'affiche
- c. *c* vaut 'a'
- d. *c* vaut 'A'

Sujet n° 10

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier pair
 - c. Affiche bonjour quand a est un entier impair
 - d. N'affiche rien (quelque soit la valeur de a)
2. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
3. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur a
 - b. l'adresse de la variable a
 - c. n'a pas de sens
 - d. la valeur de a tout simplement
4. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à $t[10]$ la valeur 45
 - b. Permettent de d'affecter à $t[10]$ la valeur 100
 - c. Permettent de d'affecter à $t[10]$ la valeur 0
 - d. Permettent de d'affecter à $t[10]$ la valeur 10
5. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. ce vers quoi pointe la variable
 - c. le contenu de la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
6. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère 'A'
7. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 62
 - c. 68
 - d. 54

8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010100110
 - 111011010101111
 - 111011010100000
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(a, b) ;`
 - `echange(*a, *b) ;`
 - `echange(&a, &b) ;`
10. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
12. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
14. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres

15. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de  $a$
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
16. `printf("%i", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
17. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - a pour valeur VRAI
  - n'a pas de valeur
  - a pour valeur FAUX
19. Après `char c; c='a'; c=c+1;`
- $c$  vaut 'b'
  - $c$  vaut 'a'
  - $c$  vaut 'A'
  - Un message d'erreur s'affiche
20. `if` est :
- Un identificateur du langage C
  - Un mot-clef du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande

# Sujet n° 11

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101111
- c. 111011010101000
- d. 111011010100000

2. `if` est :

- a. Un identificateur du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un mot-clef du langage C
- d. Un opérateur du langage C

3. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 54
- c. 62
- d. 58

4. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

5. L'expression `a<=1`

- a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- b. Réalise une affectation
- c. Diminue de 1 la valeur de `a`
- d. Utilise les opérateurs `<` et `=`

6. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut `'A'`
- b. Un message d'erreur s'affiche
- c. `c` vaut `'a'`
- d. `c` vaut `'b'`

7. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont afficher 10 nombres  
b. vont afficher 11 nombres  
c. vont afficher 9 nombres  
d. vont boucler indéfiniment
8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :  
a. n'a pas de sens  
b. la valeur de `a` tout simplement  
c. l'adresse de la variable `a`  
d. ce vers quoi pointe le pointeur `a`
9. `if (a<5) printf("Bonjour") ; a=a+1 ;`  
a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`  
b. Augmente la valeur de `a` quelque soit `a`  
c. Affiche bonjour quelque soit `a`  
d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?  
a. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`  
b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`  
c. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`  
d. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
11. L'expression `(0 == 7%3) || (1 == 9%3)`  
a. a pour valeur FAUX  
b. entraîne l'affichage d'un message d'erreur  
c. n'a pas de valeur  
d. a pour valeur VRAI
12. `printf("%i", 'A')`  
a. Affiche le caractère 'A'  
b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`  
c. Affiche le code ASCII du caractère stocké dans la variable `A`  
d. Affiche le code ASCII du caractère 'A'
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :  
a. n'a pas de sens  
b. ce vers quoi pointe le pointeur `a`  
c. la valeur de `a` tout simplement  
d. l'adresse de la variable `a`
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :  
a. `echange(*a, *b) ;`  
b. `echange(a, b) ;`  
c. `echange(&a, &b) ;`  
d. `echange(a++, b++) ;`

15. `printf("%c", 'A')`
- Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
16. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
17. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier impair
  - Affiche bonjour quand *a* est un entier pair
  - N'affiche rien (quelque soit la valeur de *a*)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
18. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
19. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier pair
  - Affiche bonjour quand *a* est un entier impair
  - N'affiche rien (quelque soit la valeur de *a*)
20. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - le contenu de la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké

# Sujet n° 12

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.



## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'A'`
  - b. Un message d'erreur s'affiche
  - c. `c` vaut `'b'`
  - d. `c` vaut `'a'`
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101000
  - c. 111011010100110
  - d. 111011010101111
3. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont boucler indéfiniment
  - b. vont afficher 10 nombres
  - c. vont afficher 9 nombres
  - d. vont afficher 11 nombres
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. ce vers quoi pointe le pointeur `a`
  - c. n'a pas de sens
  - d. l'adresse de la variable `a`
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a, b) ;`
  - b. `echange(&a, &b) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(a++, b++) ;`
6. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`

7. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
9. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de  $a$
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
10. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 58
  - 68
12. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur VRAI
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
14. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère `'A'`

15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
16. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
17. `if` est :
- Un identificateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
18. L'adresse d'une variable c'est :
- le contenu de la variable
  - ce vers quoi pointe la variable
  - l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
19. `if (a%2 = 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
20. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`

# Sujet n° 13

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Affiche bonjour quand  $a$  est un entier pair
2. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
  - a. vont afficher 10 nombres
  - b. vont afficher 9 nombres
  - c. vont boucler indéfiniment
  - d. vont afficher 11 nombres
3. L'expression `a<=1`
  - a. Diminue de 1 la valeur de  $a$
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. l'adresse de la variable `a`
  - c. la valeur de `a` tout simplement
  - d. n'a pas de sens
5. `if (a%2 = 0) printf("bonjour");`
  - a. Affiche bonjour quand  $a$  est un entier pair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier impair
6. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. le contenu de la variable
  - c. ce vers quoi pointe la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké

7. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

8. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable *A*
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100110
- c. 111011010100000
- d. 111011010101000

10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(a++, b++);`
- c. `echange(a, b);`
- d. `echange(*a, *b);`

11. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 45

12. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 54
- b. 68
- c. 58
- d. 62

13. Après `char c; c='a'; c=c+1;`

- a. *c* vaut 'a'
- b. *c* vaut 'A'
- c. *c* vaut 'b'
- d. Un message d'erreur s'affiche

14. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le caractère 'A'



15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. n'a pas de valeur
  - b. a pour valeur VRAI
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur FAUX
16. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - b. Affiche bonjour quelque soit *a*
  - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - d. Augmente la valeur de *a* quelque soit *a*
17. `if` est :
- a. Un mot-clef du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un opérateur du langage C
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
  - b. l'adresse de la variable `a`
  - c. la valeur de `a` tout simplement
  - d. ce vers quoi pointe le pointeur `a`
19. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`
20. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le caractère `'A'`

# Sujet n° 14

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'A'`
  - c. `c` vaut `'b'`
  - d. `c` vaut `'a'`
2. `printf("%c", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
3. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. l'adresse de la variable `a`
  - c. ce vers quoi pointe le pointeur `a`
  - d. n'a pas de sens
4. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier pair
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de `a`)
5. `if` est :
  - a. Un mot-clef du langage C
  - b. Un opérateur du langage C
  - c. Un identificateur du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
6. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 68
  - c. 62
  - d. 54

7. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont afficher 11 nombres  
b. vont afficher 10 nombres  
c. vont afficher 9 nombres  
d. vont boucler indéfiniment
8. `printf("%i", A)`  
a. Affiche le caractère 'A'  
b. Affiche le code ASCII du caractère stocké dans la variable A  
c. Affiche le caractère dont le code ASCII est stocké dans la variable A  
d. Affiche le code ASCII du caractère 'A'
9. `if (a<5) printf("Bonjour") ; a=a+1 ;`  
a. Affiche bonjour quelque soit a  
b. Augmente la valeur de a quelque soit a  
c. Affiche bonjour et augmente la valeur de a quelque soit a  
d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
10. `printf("%c", A)`  
a. Affiche le caractère dont le code ASCII est stocké dans la variable A  
b. Affiche le code ASCII du caractère 'A'  
c. Affiche le code ASCII du caractère stocké dans la variable A  
d. Affiche le caractère 'A'
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?  
a. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`  
b. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`  
c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`  
d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
12. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`  
a. Permettent de d'affecter à `t[10]` la valeur 45  
b. Permettent de d'affecter à `t[10]` la valeur 100  
c. Permettent de d'affecter à `t[10]` la valeur 0  
d. Permettent de d'affecter à `t[10]` la valeur 10
13. L'adresse d'une variable c'est :  
a. l'adresse d'un pointeur sur la variable  
b. ce vers quoi pointe la variable  
c. le numéro de la case mémoire où le contenu de la variable est stocké  
d. le contenu de la variable
14. L'expression `(0 == 7%3) || (1 == 9%3)`  
a. a pour valeur FAUX  
b. a pour valeur VRAI  
c. n'a pas de valeur  
d. entraîne l'affichage d'un message d'erreur

15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
16. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
17. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
  - Utilise les opérateurs `<` et `=`
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010100110
  - 111011010101000
  - 111011010101111
19. `printf("%i", 'A')`
- Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b);`
  - `echange(a++, b++);`
  - `echange(*a, *b);`
  - `echange(&a, &b);`

# Sujet n° 15

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 68
  - c. 62
  - d. 54
2. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. le contenu de la variable
  - c. ce vers quoi pointe la variable
  - d. l'adresse d'un pointeur sur la variable
3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 100
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
5. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère '`A`'
  - b. Affiche le caractère '`A`'
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
6. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. a pour valeur FAUX
7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`

8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101000
  - 111011010101111
  - 111011010100110
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
11. `if` est :
- Un opérateur du langage C
  - Un mot-clef du langage C
  - Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
12. L'expression `a<=1`
- Réalise une affectation
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Diminue de 1 la valeur de  $a$
13. `printf("%c", 'A')`
- Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
14. Après `char c; c='a'; c=c+1;`
- $c$  vaut 'A'
  - $c$  vaut 'a'
  - $c$  vaut 'b'
  - Un message d'erreur s'affiche
15. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de  $a$ )
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - Affiche bonjour quand  $a$  est un entier pair

16. `printf("%i", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
17. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```
- vont afficher 10 nombres
 - vont afficher 11 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
18. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - Affiche bonjour quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*
19. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a++, b++) ;`
 - `echange(a, b) ;`
 - `echange(&a, &b) ;`
 - `echange(*a, *b) ;`

Sujet n° 16

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 62
 - c. 54
 - d. 58
2. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 100
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. n'a pas de valeur
 - b. a pour valeur FAUX
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur VRAI
4. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - b. Affiche bonjour quelque soit *a*
 - c. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - d. Augmente la valeur de *a* quelque soit *a*
5. On suppose que **a** a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de **a** tout simplement
 - b. l'adresse de la variable **a**
 - c. ce vers quoi pointe le pointeur **a**
 - d. n'a pas de sens
6. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
7. On suppose que **a** a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de **a** tout simplement
 - b. l'adresse de la variable **a**
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur **a**

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(a++, b++);`
- c. `echange(*a, *b);`
- d. `echange(a, b);`

9. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier pair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand a est un entier impair
- d. N'affiche rien (quelque soit la valeur de a)

10. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable A

11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100000
- c. 111011010101111
- d. 111011010100110

12. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. le contenu de la variable
- c. ce vers quoi pointe la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

13. L'expression `a<=1`

- a. Diminue de 1 la valeur de a
- b. Réalise une affectation
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Utilise les opérateurs `<` et `=`

14. `if` est :

- a. Un opérateur du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un identificateur du langage C
- d. Un mot-clef du langage C

15. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de a)
- c. Affiche bonjour quand a est un entier impair
- d. Affiche bonjour quand a est un entier pair

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

17. `printf("%i", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable A
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le code ASCII du caractère 'A'

18. Après `char c; c='a'; c=c+1;`

- a. c vaut 'a'
- b. c vaut 'b'
- c. Un message d'erreur s'affiche
- d. c vaut 'A'

19. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 9 nombres
- d. vont afficher 10 nombres

20. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable A
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le code ASCII du caractère 'A'

Sujet n° 17

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 11 nombres
 - b. vont afficher 9 nombres
 - c. vont boucler indéfiniment
 - d. vont afficher 10 nombres
2. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le contenu de la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. ce vers quoi pointe la variable
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
4. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur FAUX
 - c. n'a pas de valeur
 - d. a pour valeur VRAI
5. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 45
6. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le caractère `'A'`

7. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut 'A'
 - Un message d'erreur s'affiche
 - `c` vaut 'b'
 - `c` vaut 'a'
8. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
 - Affiche bonjour quand a est un entier impair
9. `if (a%2 != 0) printf("bonjour") ;`
- Affiche bonjour quand a est un entier impair
 - Affiche bonjour quand a est un entier pair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
11. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de a
 - Réalise une affectation
12. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C
 - Un opérateur du langage C
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 62
 - 68
 - 58
14. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010101000
 - 111011010101111
 - 111011010100110

15. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
16. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
18. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(*a, *b) ;`
 - `echange(a, b) ;`
20. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'

Sujet n° 18

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
2. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
3. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Un identificateur du langage C
4. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
5. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier impair
6. On suppose que *a* a été déclarée par `int *a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur *a*
 - b. la valeur de *a* tout simplement
 - c. n'a pas de sens
 - d. l'adresse de la variable *a*
7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
 - a. `echange(*a, *b);`
 - b. `echange(a, b);`
 - c. `echange(a++, b++);`
 - d. `echange(&a, &b);`

8. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont boucler indéfiniment
b. vont afficher 11 nombres
c. vont afficher 9 nombres
d. vont afficher 10 nombres
9. `if (a<5) printf("Bonjour") ; a=a+1 ;`
a. Affiche bonjour quelque soit a
b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
c. Augmente la valeur de a quelque soit a
d. Affiche bonjour et augmente la valeur de a quelque soit a
10. L'expression `a<=1`
a. Diminue de 1 la valeur de a
b. Utilise les opérateurs `<` et `=`
c. Réalise une affectation
d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
11. `printf("%c", 'A')`
a. Affiche le caractère 'A'
b. Affiche le caractère dont le code ASCII est stocké dans la variable A
c. Affiche le code ASCII du caractère stocké dans la variable A
d. Affiche le code ASCII du caractère 'A'
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
a. 111011010100110
b. 111011010101000
c. 111011010100000
d. 111011010101111
13. `if (a%2 == 0) printf("bonjour") ;`
a. Affiche bonjour quand a est un entier impair
b. N'affiche rien (quelque soit la valeur de a)
c. Affiche bonjour quand a est un entier pair
d. Déclenche le message d'erreur `invalid lvalue in assignment`
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
a. 68
b. 62
c. 58
d. 54
15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
a. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
b. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
c. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
d. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
17. Après `char c ; c='a' ; c=c+1 ;`
- a. Un message d'erreur s'affiche
 - b. `c` vaut `'a'`
 - c. `c` vaut `'b'`
 - d. `c` vaut `'A'`
18. `printf("%i", A)`
- a. Affiche le caractère `'A'`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur VRAI
 - d. a pour valeur FAUX
20. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 0

Sujet n° 19

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 45
2. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur VRAI
 - c. n'a pas de valeur
 - d. a pour valeur FAUX
3. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. la valeur de `a` tout simplement
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
4. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 62
 - c. 54
 - d. 58
5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010100110
 - c. 111011010101111
 - d. 111011010101000
6. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère '`A`'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère '`A`'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
7. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable

8. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
9. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
 - c vaut 'b'
 - c vaut 'a'
 - c vaut 'A'
10. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de a
 - Réalise une affectation
11. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
12. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
13. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables a et b on doit écrire :
- `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
15. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - la valeur de a tout simplement
 - ce vers quoi pointe le pointeur a
 - l'adresse de la variable a

16. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C
17. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
18. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
19. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
 - Affiche bonjour quand a est un entier impair
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

Sujet n° 20

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101111
- c. 111011010101000
- d. 111011010100000

2. `printf("%c", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable A
- c. Affiche le caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

3. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Augmente la valeur de a quelque soit a
- b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- c. Affiche bonjour quelque soit a
- d. Affiche bonjour et augmente la valeur de a quelque soit a

4. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Réalise une affectation
- c. Diminue de 1 la valeur de a
- d. Utilise les opérateurs `<` et `=`

5. `printf("%i", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable A
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le code ASCII du caractère 'A'

6. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier impair
- b. N'affiche rien (quelque soit la valeur de a)
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand a est un entier pair

7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable `a`
- b. n'a pas de sens
- c. la valeur de `a` tout simplement
- d. ce vers quoi pointe le pointeur `a`

8. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
 - vont afficher 11 nombres
9. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
10. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 62
 - 58
 - 68
13. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un mot-clef du langage C
 - Un opérateur du langage C
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
15. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké

16. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - a pour valeur FAUX
 - n'a pas de valeur
17. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier pair
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
19. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
20. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
 - c* vaut 'A'
 - c* vaut 'a'
 - c* vaut 'b'

Sujet n° 21

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens
2. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :
 - a. `111011010101111`
 - b. `111011010100110`
 - c. `111011010100000`
 - d. `111011010101000`
3. `if` est :
 - a. Un opérateur du langage C
 - b. Un mot-clef du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un identificateur du langage C
4. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand `a` est un entier impair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier pair
5. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`
6. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. n'a pas de valeur
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur VRAI
7. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. le contenu de la variable

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - n'a pas de sens
 - la valeur de `a` tout simplement
9. `printf("%c", A)`
- Affiche le code ASCII du caractère '`A`'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère '`A`'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 58
 - 68
 - 62
12. L'expression `a<=1`
- Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Diminue de 1 la valeur de `a`
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b);`
 - `echange(&a, &b);`
 - `echange(a, b);`
 - `echange(a++, b++);`
14. Les instructions
- ```
i=0;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
  - vont afficher 11 nombres
15. Après `char c; c='a'; c=c+1;`
- `c` vaut '`A`'
  - `c` vaut '`b`'
  - `c` vaut '`a`'
  - Un message d'erreur s'affiche



16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Affiche bonjour quand  $a$  est un entier pair
17. `printf("%i", A)`
- a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
18. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Augmente la valeur de  $a$  quelque soit  $a$
19. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$
20. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à  $t[10]$  la valeur 10
  - b. Permettent de d'affecter à  $t[10]$  la valeur 45
  - c. Permettent de d'affecter à  $t[10]$  la valeur 0
  - d. Permettent de d'affecter à  $t[10]$  la valeur 100

# Sujet n° 22

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
2. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
3. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
4. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont afficher 11 nombres
  - b. vont afficher 9 nombres
  - c. vont afficher 10 nombres
  - d. vont boucler indéfiniment
5. `if` est :
  - a. Un opérateur du langage C
  - b. Un identificateur du langage C
  - c. Un mot-clef du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
6. Après `char c ; c='a' ; c=c+1 ;`
  - a. *c* vaut 'a'
  - b. *c* vaut 'A'
  - c. *c* vaut 'b'
  - d. Un message d'erreur s'affiche

7. L'expression `a<=1`
  - a. Réalise une affectation
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Diminue de 1 la valeur de  $a$
8. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 62
  - c. 54
  - d. 58
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. la valeur de `a` tout simplement
  - c. l'adresse de la variable `a`
  - d. n'a pas de sens
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
12. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur VRAI
  - c. n'a pas de valeur
  - d. a pour valeur FAUX
13. `if (a%2 == 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. Affiche bonjour quand  $a$  est un entier impair
14. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère 'A'

15. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. Affiche bonjour quand  $a$  est un entier impair
16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
  - b. la valeur de `a` tout simplement
  - c. n'a pas de sens
  - d. l'adresse de la variable `a`
17. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Augmente la valeur de  $a$  quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour quelque soit  $a$
  - d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
18. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 100
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 0
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b);`
  - b. `echange(a++, b++);`
  - c. `echange(&a, &b);`
  - d. `echange(*a, *b);`
20. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :
- a. `111011010101000`
  - b. `111011010100110`
  - c. `111011010101111`
  - d. `111011010100000`

# Sujet n° 23

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
2. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
3. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
4. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de *a* tout simplement
  - c. l'adresse de la variable *a*
  - d. ce vers quoi pointe le pointeur *a*
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 62
  - c. 68
  - d. 54
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
  - a. `echange(a, b);`
  - b. `echange(*a, *b);`
  - c. `echange(&a, &b);`
  - d. `echange(a++, b++);`
7. `if` est :
  - a. Un identificateur du langage C
  - b. Un mot-clef du langage C
  - c. Un opérateur du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande

8. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont boucler indéfiniment  
b. vont afficher 9 nombres  
c. vont afficher 11 nombres  
d. vont afficher 10 nombres
9. `if (a%2 == 0) printf("bonjour") ;`  
a. Déclenche le message d'erreur `invalid lvalue in assignment`  
b. Affiche bonjour quand  $a$  est un entier impair  
c. Affiche bonjour quand  $a$  est un entier pair  
d. N'affiche rien (quelque soit la valeur de  $a$ )
10. L'adresse d'une variable c'est :  
a. le contenu de la variable  
b. le numéro de la case mémoire où le contenu de la variable est stocké  
c. ce vers quoi pointe la variable  
d. l'adresse d'un pointeur sur la variable
11. L'expression `a<=1`  
a. Utilise les opérateurs `<` et `=`  
b. Diminue de 1 la valeur de  $a$   
c. Réalise une affectation  
d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :  
a. 111011010100000  
b. 111011010101111  
c. 111011010101000  
d. 111011010100110
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :  
a. n'a pas de sens  
b. l'adresse de la variable `a`  
c. la valeur de `a` tout simplement  
d. ce vers quoi pointe le pointeur `a`
14. `if (a%2 == 0) printf("bonjour") ;`  
a. N'affiche rien (quelque soit la valeur de  $a$ )  
b. Affiche bonjour quand  $a$  est un entier impair  
c. Déclenche le message d'erreur `invalid lvalue in assignment`  
d. Affiche bonjour quand  $a$  est un entier pair
15. Après `char c ; c='a' ; c=c+1 ;`  
a. Un message d'erreur s'affiche  
b. `c` vaut `'A'`  
c. `c` vaut `'b'`  
d. `c` vaut `'a'`

16. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère 'A'
17. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour quelque soit *a*
  - Augmente la valeur de *a* quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - a pour valeur VRAI
  - n'a pas de valeur
  - a pour valeur FAUX
19. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

# Sujet n° 24

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. n'a pas de sens
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

2. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère `'A'`

3. `printf("%c", A)`

- a. Affiche le code ASCII du caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

4. `printf("%i", A)`

- a. Affiche le caractère `'A'`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

5. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de `a`)
- b. Affiche bonjour quand `a` est un entier pair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand `a` est un entier impair

6. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère `'A'`

7. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :

- a. `111011010101111`
- b. `111011010100000`
- c. `111011010100110`
- d. `111011010101000`

8. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
10. L'expression `a<=1`
- Réalise une affectation
  - Diminue de 1 la valeur de  $a$
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b);`
  - `echange(a++, b++);`
  - `echange(*a, *b);`
  - `echange(a, b);`
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
13. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Déclenche le message d'erreur `invalid lvalue in assignment`
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 54
  - 68
  - 58
15. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
  - Un mot-clef du langage C
  - Un opérateur du langage C

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

17. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur VRAI
- c. n'a pas de valeur
- d. a pour valeur FAUX

18. L'adresse d'une variable c'est :

- a. le numéro de la case mémoire où le contenu de la variable est stocké
- b. ce vers quoi pointe la variable
- c. l'adresse d'un pointeur sur la variable
- d. le contenu de la variable

19. Après `char c; c='a'; c=c+1;`

- a. Un message d'erreur s'affiche
- b. `c` vaut 'A'
- c. `c` vaut 'a'
- d. `c` vaut 'b'

20. Les instructions `i=0;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 10 nombres
- c. vont afficher 11 nombres
- d. vont afficher 9 nombres



# Sujet n° 25

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'A'`
  - b. `c` vaut `'a'`
  - c. `c` vaut `'b'`
  - d. Un message d'erreur s'affiche
2. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de `a` tout simplement
  - c. l'adresse de la variable `a`
  - d. ce vers quoi pointe le pointeur `a`
3. `printf("%c", 'A')`
  - a. Affiche le caractère `'A'`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(&a, &b) ;`
  - b. `echange(a, b) ;`
  - c. `echange(a++, b++) ;`
  - d. `echange(*a, *b) ;`
5. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
6. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 100
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 10
7. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

8. L'expression `a<=1`
- Diminue de 1 la valeur de  $a$
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 68
  - 58
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
11. Les instructions
- ```
i=0 ;
while(i<10)
    printf("%i ", i);
    i++ ;
```
- vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
 - vont afficher 10 nombres
12. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le contenu de la variable
13. `if (a<5) printf("Bonjour"); a=a+1 ;`
- N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Augmente la valeur de a quelque soit a
14. `if` est :
- Un mot-clef du langage C
 - Un identificateur du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur VRAI

16. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère 'A'
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010101000
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
19. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand *a* est un entier pair
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand *a* est un entier impair
20. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*

Sujet n° 26

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
 - a. `echange(a, b) ;`
 - b. `echange(*a, *b) ;`
 - c. `echange(&a, &b) ;`
 - d. `echange(a++, b++) ;`
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 54
 - b. 62
 - c. 68
 - d. 58
4. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un opérateur du langage C
 - c. Un identificateur du langage C
 - d. Un mot-clef du langage C
5. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 0
6. `printf("%c", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
7. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Diminue de 1 la valeur de *a*
 - d. Réalise une affectation

8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010101111
 - 111011010100000
 - 111011010100110
9. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le contenu de la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
10. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
12. Après `char c; c='a'; c=c+1;`
- c vaut 'a'
 - c vaut 'A'
 - c vaut 'b'
 - Un message d'erreur s'affiche
13. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
 - Affiche bonjour quand a est un entier pair
14. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - n'a pas de valeur
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI

16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de a)
 - d. Affiche bonjour quand a est un entier pair
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. n'a pas de sens
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
19. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de a quelque soit a
 - b. Affiche bonjour quelque soit a
 - c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - d. Augmente la valeur de a quelque soit a
20. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i);
 i++;
```
- a. vont boucler indéfiniment
  - b. vont afficher 11 nombres
  - c. vont afficher 10 nombres
  - d. vont afficher 9 nombres

# Sujet n° 27

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 58
  - c. 62
  - d. 68
2. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Affiche bonjour quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Augmente la valeur de  $a$  quelque soit  $a$
3. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
  - a. `echange(&a, &b);`
  - b. `echange(*a, *b);`
  - c. `echange(a++, b++);`
  - d. `echange(a, b);`
5. `if` est :
  - a. Un opérateur du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un mot-clef du langage C
6. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 100
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 45
7. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. l'adresse d'un pointeur sur la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. le contenu de la variable

8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010101000
  - 111011010101111
  - 111011010100000
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
10. Après `char c; c='a'; c=c+1;`
- `c` vaut 'A'
  - `c` vaut 'a'
  - `c` vaut 'b'
  - Un message d'erreur s'affiche
11. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
12. `printf("%i", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
13. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
14. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
15. Les instructions `i=0;`
- ```

while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
 - vont afficher 9 nombres

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
 - b. ce vers quoi pointe le pointeur `a`
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
17. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. n'a pas de sens
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
 - b. n'a pas de valeur
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur VRAI
19. L'expression `a<=1`
- a. Diminue de 1 la valeur de `a`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Utilise les opérateurs `<` et `=`
20. `printf("%i", A)`
- a. Affiche le caractère '`A`'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère '`A`'

Sujet n° 28

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier pair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier impair
2. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
3. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. l'adresse d'un pointeur sur la variable
 - d. le contenu de la variable
4. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables a et b on doit écrire :
 - a. `echange(a++, b++) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a, b) ;`
 - d. `echange(*a, *b) ;`
6. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable a
 - b. la valeur de a tout simplement
 - c. ce vers quoi pointe le pointeur a
 - d. n'a pas de sens
7. Après `char c ; c='a' ; c=c+1 ;`
 - a. c vaut 'A'
 - b. c vaut 'b'
 - c. Un message d'erreur s'affiche
 - d. c vaut 'a'

8. `if` est :
- Un identificateur du langage C
 - Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur VRAI
 - a pour valeur FAUX
10. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 54
 - 58
 - 68
12. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
13. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
14. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
17. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le caractère 'A'
18. L'expression `a<=1`
- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Utilise les opérateurs `<` et `=`
 - c. Diminue de 1 la valeur de *a*
 - d. Réalise une affectation
19. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable *a*
 - b. la valeur de *a* tout simplement
 - c. ce vers quoi pointe le pointeur *a*
 - d. n'a pas de sens
20. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand *a* est un entier pair
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand *a* est un entier impair

Sujet n° 29

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère `'A'`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

3. `if` est :

- a. Un opérateur du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un mot-clef du langage C
- d. Un identificateur du langage C

4. `printf("%c", A)`

- a. Affiche le caractère `'A'`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. n'a pas de sens
- c. l'adresse de la variable `a`
- d. la valeur de `a` tout simplement

6. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 100

7. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier impair
- d. Affiche bonjour quand `a` est un entier pair

8. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
10. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
 - `echange(a, b) ;`
 - `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
12. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
 - `c` vaut 'A'
 - `c` vaut 'b'
 - `c` vaut 'a'
13. L'expression `a<=1`
- Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Diminue de 1 la valeur de `a`
14. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable
15. Les instructions `i=0 ;`
- ```

while(i<10)
 printf("%i ", i) ;
 i++ ;

```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment



16. `if (a<5) printf("Bonjour"); a=a+1;`  
a. Affiche bonjour quelque soit  $a$   
b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$   
c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$   
d. Augmente la valeur de  $a$  quelque soit  $a$
17. L'expression `(0 == 7%3) || (1 == 9%3)`  
a. a pour valeur VRAI  
b. entraîne l'affichage d'un message d'erreur  
c. a pour valeur FAUX  
d. n'a pas de valeur
18. `if (a%2 == 0) printf("bonjour");`  
a. Affiche bonjour quand  $a$  est un entier impair  
b. N'affiche rien (quelque soit la valeur de  $a$ )  
c. Affiche bonjour quand  $a$  est un entier pair  
d. Déclenche le message d'erreur `invalid lvalue in assignment`
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :  
a. 111011010101000  
b. 111011010100110  
c. 111011010101111  
d. 111011010100000
20. Le nombre qui se note 110110 en base 2 se note en base 10 :  
a. 62  
b. 58  
c. 68  
d. 54

# Sujet n° 30

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Augmente la valeur de  $a$  quelque soit  $a$
  - b. Affiche bonjour quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
2. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
4. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le caractère 'A'
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
  - a. `echange(a++, b++) ;`
  - b. `echange(&a, &b) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(a, b) ;`
6. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
7. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b.  $c$  vaut 'a'
  - c.  $c$  vaut 'b'
  - d.  $c$  vaut 'A'

8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

10. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 58
- c. 54
- d. 62

11. `if` est :

- a. Un opérateur du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un mot-clef du langage C
- d. Un identificateur du langage C

12. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur VRAI
- c. a pour valeur FAUX
- d. n'a pas de valeur

13. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 11 nombres
- c. vont boucler indéfiniment
- d. vont afficher 10 nombres

14. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 10

16. L'expression `a<=1`
- a. Diminue de 1 la valeur de  $a$
  - b. Utilise les opérateurs `<` et `=`
  - c. Réalise une affectation
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
17. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100000
  - b. 111011010101000
  - c. 111011010100110
  - d. 111011010101111
20. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. le contenu de la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. ce vers quoi pointe la variable

# Sujet n° 31

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. En cours, on a vu comment à l'aide de pointeurs définir une fonction **echange** qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables **a** et **b** on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a, b) ;`
- c. `echange(*a, *b) ;`
- d. `echange(a++, b++) ;`

2. `printf("%i", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable *A*
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

3. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère 'A'
- d. Affiche le code ASCII du caractère stocké dans la variable *A*

4. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Augmente la valeur de *a* quelque soit *a*
- b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
- c. Affiche bonjour quelque soit *a*
- d. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*

5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101000
- c. 111011010100000
- d. 111011010101111

6. On suppose que **a** a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. ce vers quoi pointe le pointeur **a**
- c. l'adresse de la variable **a**
- d. la valeur de **a** tout simplement

7. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
9. L'expression `a<=1`
- Diminue de 1 la valeur de *a*
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
10. Les instructions
- ```
i=0 ;
while(i<10)
    printf("%i ", i) ;
    i++ ;
```
- vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
12. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le contenu de la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
13. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
 - Un mot-clef du langage C
 - Un identificateur du langage C
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 58
 - 68
 - 62

15. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Affiche bonjour quand a est un entier impair
 - d. Affiche bonjour quand a est un entier pair
16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur FAUX
 - d. n'a pas de valeur
17. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère 'A'
18. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
 - b. c vaut 'A'
 - c. c vaut 'a'
 - d. c vaut 'b'
19. `if (a%2 = 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier pair
20. `printf("%c", 'A')`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A

Sujet n° 32

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 0
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(&a, &b) ;`
 - b. `echange(a, b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(a++, b++) ;`
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. n'a pas de valeur
 - d. a pour valeur FAUX
4. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. l'adresse d'un pointeur sur la variable
 - d. le contenu de la variable
5. `if (a%2 == 0) printf("bonjour") ;`
 - a. N'affiche rien (quelque soit la valeur de `a`)
 - b. Affiche bonjour quand `a` est un entier impair
 - c. Affiche bonjour quand `a` est un entier pair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
6. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère `'A'`
7. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère `'A'`

8. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
 - 62
 - 54
 - 58
10. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un mot-clef du langage C
11. Les instructions `i=0;`
- ```

while(i<10)
 printf("%i ", i);
 i++;

```
- vont afficher 11 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
  - vont afficher 9 nombres
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
13. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

16. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
  - Un message d'erreur s'affiche
  - `c` vaut `'A'`
  - `c` vaut `'b'`
17. `printf("%i", 'A')`
- Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
18. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010101000
  - 111011010100000
  - 111011010100110
20. L'expression `a<=1`
- A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de `a`
  - Réalise une affectation



# Sujet n° 33

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 9 nombres
- c. vont afficher 10 nombres
- d. vont afficher 11 nombres

2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

3. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur VRAI
- c. a pour valeur FAUX
- d. entraîne l'affichage d'un message d'erreur

4. `if` est :

- a. Un opérateur du langage C
- b. Un identificateur du langage C
- c. Un mot-clef du langage C
- d. Une commande qu'on tape dans la fenêtre de commande

5. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit  $a$
- b. Augmente la valeur de  $a$  quelque soit  $a$
- c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
- d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$

6. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 0
- c. Permettent de d'affecter à `t[10]` la valeur 100
- d. Permettent de d'affecter à `t[10]` la valeur 10

7. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
  - `c` vaut `'A'`
  - Un message d'erreur s'affiche
  - `c` vaut `'b'`
8. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
10. `printf("%c", A)`
- Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
  - `echange(a++, b++) ;`
  - `echange(a, b) ;`
  - `echange(*a, *b) ;`
12. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier impair
13. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
14. `if (a%2 = 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`

15. `printf("%i", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100110
  - 111011010100000
  - 111011010101000
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
  - 58
  - 54
  - 62
18. L'expression `a<=1`
- Réalise une affectation
  - Diminue de 1 la valeur de *a*
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
19. L'adresse d'une variable c'est :
- le contenu de la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
  - l'adresse d'un pointeur sur la variable
20. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - n'a pas de sens
  - l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`

# Sujet n° 34

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont afficher 9 nombres
  - b. vont afficher 11 nombres
  - c. vont boucler indéfiniment
  - d. vont afficher 10 nombres
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. n'a pas de valeur
  - b. a pour valeur VRAI
  - c. a pour valeur FAUX
  - d. entraîne l'affichage d'un message d'erreur
3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a, b) ;`
  - b. `echange(a++, b++) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(&a, &b) ;`
4. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`
5. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Affiche bonjour quand `a` est un entier pair
6. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`



7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. l'adresse de la variable `a`
  - c. n'a pas de sens
  - d. la valeur de `a` tout simplement
8. `if` est :
  - a. Un opérateur du langage C
  - b. Un identificateur du langage C
  - c. Un mot-clef du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101000
  - b. 111011010100000
  - c. 111011010100110
  - d. 111011010101111
10. L'expression `a<=1`
  - a. Réalise une affectation
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Diminue de 1 la valeur de `a`
11. `printf("%c", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
12. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable
13. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Augmente la valeur de `a` quelque soit `a`
  - b. Affiche bonjour quelque soit `a`
  - c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
14. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand `a` est un entier pair
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier impair

15. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
  - Un message d'erreur s'affiche
  - `c` vaut `'b'`
  - `c` vaut `'A'`
16. `printf("%i", A)`
- Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 68
  - 58
  - 54
18. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
19. `printf("%i", 'A')`
- Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
20. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens

# Sujet n° 35

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
  - a. Un mot-clef du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un opérateur du langage C
2. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - c. Affiche le code ASCII du caractère stocké dans la variable A
  - d. Affiche le caractère 'A'
3. `printf("%i", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le code ASCII du caractère stocké dans la variable A
4. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
5. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
6. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont afficher 11 nombres
  - b. vont afficher 9 nombres
  - c. vont afficher 10 nombres
  - d. vont boucler indéfiniment
7. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de a)
  - b. Affiche bonjour quand a est un entier impair
  - c. Affiche bonjour quand a est un entier pair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`

8. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - a pour valeur VRAI
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
  - `echange(a++, b++) ;`
  - `echange(a, b) ;`
  - `echange(*a, *b) ;`
10. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
11. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
12. `printf("%c", A)`
- Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
14. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - `c` vaut 'A'
  - `c` vaut 'b'
  - `c` vaut 'a'
15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0

16. L'expression `a<=1`
- Diminue de 1 la valeur de  $a$
  - Réalise une affectation
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 62
  - 54
  - 68
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010100000
  - 111011010101000
  - 111011010101111
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

# Sujet n° 36

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.



## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le caractère 'A'
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101000
  - b. 111011010101111
  - c. 111011010100110
  - d. 111011010100000
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 58
  - c. 68
  - d. 62
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a++, b++) ;`
  - b. `echange(*a, *b) ;`
  - c. `echange(a, b) ;`
  - d. `echange(&a, &b) ;`
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echage(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echage(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echage(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echage(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
6. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - c. Affiche le code ASCII du caractère stocké dans la variable A
  - d. Affiche le caractère 'A'

7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
8. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier pair
9. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - c* vaut 'A'
  - c* vaut 'a'
  - c* vaut 'b'
10. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur VRAI
11. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour quelque soit *a*
  - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
12. L'expression `a<=1`
- Réalise une affectation
  - Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de *a*
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
13. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - l'adresse de la variable *a*
  - la valeur de *a* tout simplement
  - ce vers quoi pointe le pointeur *a*
14. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
15. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
  - l'adresse de la variable *a*
  - la valeur de *a* tout simplement
  - n'a pas de sens

16. `if` est :
- a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un mot-clef du langage C
  - c. Un opérateur du langage C
  - d. Un identificateur du langage C
17. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand *a* est un entier pair
  - b. N'affiche rien (quelque soit la valeur de *a*)
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand *a* est un entier impair
18. L'adresse d'une variable c'est :
- a. le contenu de la variable
  - b. l'adresse d'un pointeur sur la variable
  - c. ce vers quoi pointe la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
19. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 0
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 45
20. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- a. vont afficher 10 nombres
  - b. vont afficher 11 nombres
  - c. vont afficher 9 nombres
  - d. vont boucler indéfiniment

# Sujet n° 37

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 58
  - c. 62
  - d. 54
2. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le caractère 'A'
3. Après `char c ; c='a' ; c=c+1 ;`
  - a. *c* vaut 'A'
  - b. Un message d'erreur s'affiche
  - c. *c* vaut 'b'
  - d. *c* vaut 'a'
4. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont afficher 10 nombres
  - b. vont afficher 11 nombres
  - c. vont boucler indéfiniment
  - d. vont afficher 9 nombres
5. L'expression `a<=1`
  - a. Utilise les opérateurs `<` et `=`
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Réalise une affectation
  - d. Diminue de 1 la valeur de *a*
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
  - a. `echange(&a, &b) ;`
  - b. `echange(*a, *b) ;`
  - c. `echange(a, b) ;`
  - d. `echange(a++, b++) ;`

7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens
  - l'adresse de la variable `a`
8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100000
  - 111011010100110
  - 111011010101000
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
12. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de `a`)
13. `printf("%i", 'A')`
- Affiche le caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
14. `if` est :
- Un mot-clef du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C



15. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
16. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour quelque soit *a*
  - b. Augmente la valeur de *a* quelque soit *a*
  - c. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - d. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
17. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
  - b. entraîne l'affichage d'un message d'erreur
  - c. n'a pas de valeur
  - d. a pour valeur FAUX
18. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 45
20. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

# Sujet n° 38

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
2. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. le contenu de la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101111
  - b. 111011010100110
  - c. 111011010101000
  - d. 111011010100000
4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
5. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
  - a. `echange(a, b);`
  - b. `echange(*a, *b);`
  - c. `echange(&a, &b);`
  - d. `echange(a++, b++);`

7. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont afficher 10 nombres  
b. vont afficher 11 nombres  
c. vont boucler indéfiniment  
d. vont afficher 9 nombres
8. `if (a<5) printf("Bonjour") ; a=a+1 ;`  
a. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$   
b. Affiche bonjour quelque soit  $a$   
c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$   
d. Augmente la valeur de  $a$  quelque soit  $a$
9. `if (a%2 == 0) printf("bonjour") ;`  
a. Affiche bonjour quand  $a$  est un entier pair  
b. Affiche bonjour quand  $a$  est un entier impair  
c. N'affiche rien (quelque soit la valeur de  $a$ )  
d. Déclenche le message d'erreur `invalid lvalue in assignment`
10. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`  
a. Permettent de d'affecter à `t[10]` la valeur 10  
b. Permettent de d'affecter à `t[10]` la valeur 45  
c. Permettent de d'affecter à `t[10]` la valeur 0  
d. Permettent de d'affecter à `t[10]` la valeur 100
11. `printf("%c", A)`  
a. Affiche le caractère 'A'  
b. Affiche le code ASCII du caractère 'A'  
c. Affiche le code ASCII du caractère stocké dans la variable  $A$   
d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
12. L'expression `a<=1`  
a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon  
b. Utilise les opérateurs `<` et `=`  
c. Diminue de 1 la valeur de  $a$   
d. Réalise une affectation
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :  
a. la valeur de `a` tout simplement  
b. l'adresse de la variable `a`  
c. n'a pas de sens  
d. ce vers quoi pointe le pointeur `a`
14. Après `char c ; c='a' ; c=c+1 ;`  
a. `c` vaut 'b'  
b. Un message d'erreur s'affiche  
c. `c` vaut 'a'  
d. `c` vaut 'A'

15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur VRAI
16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
  - b. 54
  - c. 68
  - d. 58
17. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
  - b. N'affiche rien (quelque soit la valeur de *a*)
  - c. Affiche bonjour quand a est un entier pair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
18. `if` est :
- a. Un identificateur du langage C
  - b. Un mot-clef du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un opérateur du langage C
19. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
20. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`

# Sujet n° 39

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est IMPÉRATIF de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :

- a. Un identificateur du langage C
- b. Un opérateur du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un mot-clef du langage C

2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(a++, b++);`
- c. `echange(a, b);`
- d. `echange(*a, *b);`

3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. n'a pas de sens
- c. l'adresse de la variable `a`
- d. la valeur de `a` tout simplement

4. `printf("%c", A)`

- a. Affiche le code ASCII du caractère '`A`'
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère '`A`'

5. Les instructions `i=0;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont boucler indéfiniment
- c. vont afficher 11 nombres
- d. vont afficher 10 nombres

6. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100000
- c. 111011010101000
- d. 111011010100110

7. L'expression `a<=1`
  - a. Diminue de 1 la valeur de *a*
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
8. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. ce vers quoi pointe la variable
  - d. l'adresse d'un pointeur sur la variable
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
10. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère '*A*'
  - c. Affiche le caractère '*A*'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
11. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère '*A*'
  - b. Affiche le caractère '*A*'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
13. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. l'adresse de la variable *a*
  - c. la valeur de *a* tout simplement
  - d. ce vers quoi pointe le pointeur *a*
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 54
  - c. 58
  - d. 68

15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. entraîne l'affichage d'un message d'erreur
16. `if (a%2 = 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
17. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
18. Après `char c; c='a'; c=c+1;`
- a.  $c$  vaut 'a'
  - b.  $c$  vaut 'A'
  - c.  $c$  vaut 'b'
  - d. Un message d'erreur s'affiche
19. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
20. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le caractère 'A'

# Sujet n° 40

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 10
2. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Réalise une affectation
  - c. Utilise les opérateurs `<` et `=`
  - d. Diminue de 1 la valeur de `a`
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. ce vers quoi pointe le pointeur `a`
  - c. n'a pas de sens
  - d. l'adresse de la variable `a`
4. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'b'`
  - c. `c` vaut `'a'`
  - d. `c` vaut `'A'`
5. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Affiche bonjour quelque soit `a`
  - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - c. Augmente la valeur de `a` quelque soit `a`
  - d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
6. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère `'A'`
7. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 58
  - c. 54
  - d. 68

8. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
9. L'adresse d'une variable c'est :
- le contenu de la variable
  - l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
11. `printf("%c", A)`
- Affiche le caractère '`A`'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
13. `printf("%c", 'A')`
- Affiche le caractère '`A`'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
14. `printf("%i", 'A')`
- Affiche le caractère '`A`'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- `a` pour valeur FAUX
  - `a` pour valeur VRAI
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur

16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a++, b++) ;`
- c. `echange(a, b) ;`
- d. `echange(*a, *b) ;`

17. `if` est :

- a. Un mot-clef du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un opérateur du langage C
- d. Un identificateur du langage C

18. `if (a%2 == 0) printf("bonjour") ;`

- a. N'affiche rien (quelque soit la valeur de *a*)
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand *a* est un entier impair
- d. Affiche bonjour quand *a* est un entier pair

19. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```

- a. vont afficher 10 nombres
- b. vont afficher 11 nombres
- c. vont boucler indéfiniment
- d. vont afficher 9 nombres

20. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010100110
- c. 111011010101111
- d. 111011010101000



# Sujet n° 41

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. l'adresse de la variable `a`
  - d. ce vers quoi pointe le pointeur `a`
2. `if (a%2 == 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Affiche bonjour quand `a` est un entier pair
3. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
  - a. vont afficher 11 nombres
  - b. vont afficher 9 nombres
  - c. vont afficher 10 nombres
  - d. vont boucler indéfiniment
4. `printf("%c", A)`
  - a. Affiche le caractère '`A`'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère '`A`'
5. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Augmente la valeur de `a` quelque soit `a`
  - b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - c. Affiche bonjour quelque soit `a`
  - d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
6. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le caractère '`A`'
  - d. Affiche le code ASCII du caractère '`A`'

7. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le contenu de la variable
8. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur FAUX
  - c. n'a pas de valeur
  - d. a pour valeur VRAI
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 58
  - c. 62
  - d. 68
10. L'expression `a<=1`
  - a. Utilise les opérateurs `<` et `=`
  - b. Diminue de 1 la valeur de  $a$
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Réalise une affectation
11. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le code ASCII du caractère 'A'
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(&a, &b);`
  - b. `echange(a++, b++);`
  - c. `echange(a, b);`
  - d. `echange(*a, *b);`
13. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

15. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
  - Un message d'erreur s'affiche
  - `c` vaut `'a'`
  - `c` vaut `'A'`
16. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101111
  - 111011010100110
  - 111011010101000
18. `if` est :
- Un identificateur du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un mot-clef du langage C
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
20. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`

# Sujet n° 42

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 11 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment

2. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Diminue de 1 la valeur de  $a$
- d. Réalise une affectation

3. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- b. Affiche le code ASCII du caractère stocké dans la variable  $A$
- c. Affiche le caractère 'A'
- d. Affiche le code ASCII du caractère 'A'

4. L'adresse d'une variable c'est :

- a. ce vers quoi pointe la variable
- b. le numéro de la case mémoire où le contenu de la variable est stocké
- c. le contenu de la variable
- d. l'adresse d'un pointeur sur la variable

5. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- c. Affiche le code ASCII du caractère stocké dans la variable  $A$
- d. Affiche le caractère 'A'

6. Après `char c ; c='a' ; c=c+1 ;`

- a.  $c$  vaut 'A'
- b.  $c$  vaut 'b'
- c.  $c$  vaut 'a'
- d. Un message d'erreur s'affiche

7. `printf("%i", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable  $A$
- c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- d. Affiche le code ASCII du caractère 'A'



8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
  - `echange(*a, *b) ;`
  - `echange(a++, b++) ;`
  - `echange(&a, &b) ;`
10. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
  - Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 58
  - 68
  - 62
12. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur VRAI
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
  - 111011010101111
  - 111011010100000
  - 111011010100110
14. `if` est :
- Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
  - Un opérateur du langage C
15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
17. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
  - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
18. `printf("%i", 'A')`
- Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
19. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

# Sujet n° 43

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
2. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 10
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101000
  - c. 111011010100110
  - d. 111011010101111
4. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. ce vers quoi pointe la variable
  - d. l'adresse d'un pointeur sur la variable
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction **echange** qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables **a** et **b** on doit écrire :
  - a. `echange(*a, *b);`
  - b. `echange(a, b);`
  - c. `echange(&a, &b);`
  - d. `echange(a++, b++);`
6. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand *a* est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de *a*)
  - d. Affiche bonjour quand *a* est un entier impair

7. L'expression `a<=1`
  - a. Réalise une affectation
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Diminue de 1 la valeur de  $a$
  - d. Utilise les opérateurs `<` et `=`
8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur `a`
  - c. l'adresse de la variable `a`
  - d. la valeur de `a` tout simplement
9. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Affiche bonjour quand  $a$  est un entier pair
10. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un opérateur du langage C
  - c. Un mot-clef du langage C
  - d. Un identificateur du langage C
11. Après `char c; c='a'; c=c+1;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'b'`
  - d. `c` vaut `'A'`
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 62
  - c. 58
  - d. 54
14. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. entraîne l'affichage d'un message d'erreur

15. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
16. `if (a<5) printf("Bonjour"); a=a+1;`
- a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - b. Affiche bonjour quelque soit `a`
  - c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - d. Augmente la valeur de `a` quelque soit `a`
17. `printf("%c", A)`
- a. Affiche le caractère '`A`'
  - b. Affiche le code ASCII du caractère '`A`'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
18. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère '`A`'
  - c. Affiche le caractère '`A`'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
19. Les instructions `i=0;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont afficher 9 nombres
 - b. vont afficher 11 nombres
 - c. vont afficher 10 nombres
 - d. vont boucler indéfiniment
20. `printf("%c", 'A')`
- a. Affiche le caractère '`A`'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère '`A`'

Sujet n° 44

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Utilise les opérateurs `<` et `=`
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de a
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010100000
 - c. 111011010100110
 - d. 111011010101000
3. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Augmente la valeur de a quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Affiche bonjour quelque soit a
 - d. Affiche bonjour et augmente la valeur de a quelque soit a
4. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`
5. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
6. Après `char c; c='a'; c=c+1;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut `'A'`
 - c. `c` vaut `'a'`
 - d. `c` vaut `'b'`
7. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
9. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
 - `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(a, b) ;`
11. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
12. L'adresse d'une variable c'est :
- le contenu de la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - l'adresse d'un pointeur sur la variable
13. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
- vont boucler indéfiniment
 - vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont afficher 10 nombres
14. `if` est :
- Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un opérateur du langage C
15. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

17. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de a)
- b. Affiche bonjour quand a est un entier impair
- c. Affiche bonjour quand a est un entier pair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

18. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur VRAI
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur FAUX

19. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 58
- c. 62
- d. 54

20. `if (a%2 = 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de a)
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand a est un entier impair
- d. Affiche bonjour quand a est un entier pair

Sujet n° 45

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
2. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100
3. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un mot-clef du langage C
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C
4. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur VRAI
 - d. n'a pas de valeur
5. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. n'a pas de sens
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`
6. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A

7. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont boucler indéfiniment
b. vont afficher 11 nombres
c. vont afficher 10 nombres
d. vont afficher 9 nombres
8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
a. 111011010100000
b. 111011010101111
c. 111011010100110
d. 111011010101000
9. `printf("%c", 'A')`
a. Affiche le caractère 'A'
b. Affiche le caractère dont le code ASCII est stocké dans la variable A
c. Affiche le code ASCII du caractère 'A'
d. Affiche le code ASCII du caractère stocké dans la variable A
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
11. L'adresse d'une variable c'est :
a. ce vers quoi pointe la variable
b. le contenu de la variable
c. le numéro de la case mémoire où le contenu de la variable est stocké
d. l'adresse d'un pointeur sur la variable
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
a. 58
b. 68
c. 54
d. 62
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
a. `echange(*a, *b) ;`
b. `echange(a++, b++) ;`
c. `echange(&a, &b) ;`
d. `echange(a, b) ;`

14. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de a
15. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Augmente la valeur de a quelque soit a
16. Après `char c; c='a'; c=c+1;`
- c vaut `'A'`
 - c vaut `'a'`
 - Un message d'erreur s'affiche
 - c vaut `'b'`
17. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère `'A'`
18. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier pair
19. On suppose que a a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable a
 - la valeur de a tout simplement
 - n'a pas de sens
 - ce vers quoi pointe le pointeur a
20. `if (a%2 = 0) printf("bonjour");`
- Affiche bonjour quand a est un entier pair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair

Sujet n° 46

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
2. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand a est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier pair
 - d. N'affiche rien (quelque soit la valeur de a)
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur FAUX
4. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier pair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier impair
5. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. Réalise une affectation
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Diminue de 1 la valeur de a
6. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`
7. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable A

8. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
9. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 10 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres
10. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
11. `if` est :
- Un identificateur du langage C
 - Un mot-clef du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
12. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
 - c* vaut 'a'
 - c* vaut 'b'
 - c* vaut 'A'
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
14. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable *a*
 - ce vers quoi pointe le pointeur *a*
 - la valeur de *a* tout simplement
15. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
 - b. 111011010101111
 - c. 111011010100110
 - d. 111011010100000
17. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. le contenu de la variable
 - d. ce vers quoi pointe la variable
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(*a, *b);`
 - b. `echange(&a, &b);`
 - c. `echange(a++, b++);`
 - d. `echange(a, b);`
20. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 100

Sujet n° 47

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 45
2. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
3. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
 - a. vont afficher 10 nombres
 - b. vont afficher 9 nombres
 - c. vont afficher 11 nombres
 - d. vont boucler indéfiniment
4. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010100110
 - c. 111011010101111
 - d. 111011010101000
6. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le caractère 'A'

7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
8. `if` est :
- Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
 - Un identificateur du langage C
9. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(*a, *b);`
 - `echange(&a, &b);`
 - `echange(a, b);`
 - `echange(a++, b++);`
11. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de *a*
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 68
 - 54
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
14. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable

15. `if (a<5) printf("Bonjour"); a=a+1;`
a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
b. Augmente la valeur de a quelque soit a
c. Affiche bonjour et augmente la valeur de a quelque soit a
d. Affiche bonjour quelque soit a
16. Après `char c; c='a'; c=c+1;`
a. c vaut 'a'
b. c vaut 'A'
c. Un message d'erreur s'affiche
d. c vaut 'b'
17. L'expression `(0 == 7%3) || (1 == 9%3)`
a. a pour valeur FAUX
b. a pour valeur VRAI
c. entraîne l'affichage d'un message d'erreur
d. n'a pas de valeur
18. `if (a%2 == 0) printf("bonjour");`
a. Déclenche le message d'erreur `invalid lvalue in assignment`
b. N'affiche rien (quelque soit la valeur de a)
c. Affiche bonjour quand a est un entier pair
d. Affiche bonjour quand a est un entier impair
19. On suppose que a a été déclarée par `int a`. L'expression `&a` a pour valeur :
a. la valeur de a tout simplement
b. l'adresse de la variable a
c. ce vers quoi pointe le pointeur a
d. n'a pas de sens
20. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
a. ce vers quoi pointe le pointeur a
b. l'adresse de la variable a
c. la valeur de a tout simplement
d. n'a pas de sens

Sujet n° 48

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
2. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Affiche bonjour quand *a* est un entier pair
 - c. Affiche bonjour quand *a* est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
3. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100
4. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le caractère 'A'
5. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Utilise les opérateurs `<` et `=`
 - c. Diminue de 1 la valeur de *a*
 - d. Réalise une affectation
6. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le contenu de la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. ce vers quoi pointe la variable
7. `if (a%2 = 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier impair

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b) ;`
- b. `echange(a++, b++) ;`
- c. `echange(*a, *b) ;`
- d. `echange(&a, &b) ;`

9. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut `'b'`
- b. Un message d'erreur s'affiche
- c. `c` vaut `'A'`
- d. `c` vaut `'a'`

10. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

12. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur VRAI
- b. a pour valeur FAUX
- c. entraîne l'affichage d'un message d'erreur
- d. n'a pas de valeur

13. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un identificateur du langage C
- c. Un opérateur du langage C
- d. Un mot-clef du langage C

14. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour quelque soit `a`
- b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- d. Augmente la valeur de `a` quelque soit `a`

15. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

- a. vont boucler indéfiniment
- b. vont afficher 10 nombres
- c. vont afficher 11 nombres
- d. vont afficher 9 nombres

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101111
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010100110
17. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. n'a pas de sens
18. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 58
 - b. 54
 - c. 62
 - d. 68
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

Sujet n° 49

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100110
- c. 111011010101111
- d. 111011010100000

2. `if (a<5) printf("Bonjour"); a=a+1;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- b. Augmente la valeur de a quelque soit a
- c. Affiche bonjour quelque soit a
- d. Affiche bonjour et augmente la valeur de a quelque soit a

3. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 0
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 100

4. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 62
- c. 58
- d. 54

5. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 9 nombres

6. Après `char c; c='a'; c=c+1;`

- a. Un message d'erreur s'affiche
- b. c vaut 'A'
- c. c vaut 'b'
- d. c vaut 'a'

7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
8. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`
9. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
10. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Réalise une affectation
 - c. Utilise les opérateurs `<` et `=`
 - d. Diminue de 1 la valeur de `a`
11. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. n'a pas de sens
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(a++, b++) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a, b) ;`
 - d. `echange(*a, *b) ;`

15. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. le contenu de la variable
 - d. ce vers quoi pointe la variable
16. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier pair
17. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. Affiche bonjour quand a est un entier pair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. N'affiche rien (quelque soit la valeur de a)
18. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`
19. `if` est :
- a. Un mot-clef du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C
20. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. a pour valeur VRAI

Sujet n° 50

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier pair
 - c. N'affiche rien (quelque soit la valeur de a)
 - d. Affiche bonjour quand a est un entier impair
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. la valeur de `a` tout simplement
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
3. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le contenu de la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. ce vers quoi pointe la variable
4. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un mot-clef du langage C
 - c. Un opérateur du langage C
 - d. Un identificateur du langage C
5. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
6. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. `A` pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Diminue de 1 la valeur de a
 - d. Réalise une affectation
7. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier pair
 - c. Affiche bonjour quand a est un entier impair
 - d. N'affiche rien (quelque soit la valeur de a)

8. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
 - `c` vaut `'a'`
 - `c` vaut `'A'`
 - Un message d'erreur s'affiche
9. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
13. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
14. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur FAUX
 - a pour valeur VRAI

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100000
- c. 111011010100110
- d. 111011010101111

17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b) ;`
- b. `echange(*a, *b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(&a, &b) ;`

18. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 10

19. `printf("%c", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère 'A'

20. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

- a. vont afficher 11 nombres
- b. vont afficher 9 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment

Sujet n° 51

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Affiche bonjour quand *a* est un entier pair
 - c. Affiche bonjour quand *a* est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
2. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. n'a pas de valeur
 - c. a pour valeur VRAI
 - d. entraîne l'affichage d'un message d'erreur
3. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable *a*
 - b. ce vers quoi pointe le pointeur *a*
 - c. la valeur de *a* tout simplement
 - d. n'a pas de sens
4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
5. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur *a*
 - b. n'a pas de sens
 - c. l'adresse de la variable *a*
 - d. la valeur de *a* tout simplement
6. Les instructions

```
i=0;
while(i<10)
    printf("%i ", i);
i++;
```

 - a. vont afficher 9 nombres
 - b. vont afficher 11 nombres
 - c. vont afficher 10 nombres
 - d. vont boucler indéfiniment

7. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
 - `c` vaut `'A'`
 - Un message d'erreur s'affiche
 - `c` vaut `'b'`
8. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
10. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le contenu de la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
11. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010101000
 - 111011010100000
 - 111011010100110
13. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
14. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`

15. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- a. `echange(a, b) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(*a, *b) ;`
17. `printf("%c", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
18. L'expression `a<=1`
- a. Utilise les opérateurs `<` et `=`
 - b. Réalise une affectation
 - c. Diminue de 1 la valeur de *a*
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
19. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Augmente la valeur de *a* quelque soit *a*
 - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - d. Affiche bonjour quelque soit *a*
20. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*

Sujet n° 52

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier pair
2. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
 - a. vont afficher 9 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 11 nombres
 - d. vont afficher 10 nombres
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens
5. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère `'A'`
6. Après `char c; c='a'; c=c+1;`
 - a. `c` vaut `'b'`
 - b. `c` vaut `'A'`
 - c. `c` vaut `'a'`
 - d. Un message d'erreur s'affiche

7. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
8. `if` est :
- Un mot-clef du langage C
 - Un identificateur du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
9. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010101111
 - 111011010100000
 - 111011010100110
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur VRAI
12. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
14. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100

15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b);`
- b. `echange(a++, b++);`
- c. `echange(&a, &b);`
- d. `echange(*a, *b);`

16. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Augmente la valeur de a quelque soit a
- b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- c. Affiche bonjour et augmente la valeur de a quelque soit a
- d. Affiche bonjour quelque soit a

17. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Réalise une affectation
- c. Diminue de 1 la valeur de a
- d. Utilise les opérateurs `<` et `=`

18. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable A
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echage(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echage(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echage(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echage(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

20. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. N'affiche rien (quelque soit la valeur de a)
- d. Affiche bonjour quand a est un entier pair

Sujet n° 53

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère stocké dans la variable A
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
3. `printf("%i", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère stocké dans la variable A
4. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand `a` est un entier impair
 - b. Affiche bonjour quand `a` est un entier pair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
5. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. l'adresse d'un pointeur sur la variable
 - d. le contenu de la variable
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(&a, &b);`
 - b. `echange(a++, b++);`
 - c. `echange(a, b);`
 - d. `echange(*a, *b);`
7. L'expression `a--`
 - a. Diminue de 1 la valeur de `a`
 - b. Réalise une affectation
 - c. Utilise les opérateurs `<` et `=`
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
9. Les instructions `i=0 ;`
- ```
while(i<10)
 printf("%i ", i);
 i++;
```
- vont boucler indéfiniment
  - vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont afficher 10 nombres
10. `if` est :
- Un identificateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - n'a pas de valeur
  - a pour valeur VRAI
  - entraîne l'affichage d'un message d'erreur
12. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
13. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
14. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
15. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'A'`
  - Un message d'erreur s'affiche
  - `c` vaut `'b'`
  - `c` vaut `'a'`



16. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Augmente la valeur de  $a$  quelque soit  $a$
  - b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
  - b. 111011010101000
  - c. 111011010100000
  - d. 111011010101111
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 68
  - b. 54
  - c. 58
  - d. 62
20. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'

# Sujet n° 54

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 54
  - c. 68
  - d. 62
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de `a` tout simplement
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
3. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'A'`
  - c. `c` vaut `'a'`
  - d. `c` vaut `'b'`
4. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un mot-clef du langage C
  - c. Un identificateur du langage C
  - d. Un opérateur du langage C
5. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
6. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Affiche bonjour quelque soit `a`
  - b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - d. Augmente la valeur de `a` quelque soit `a`
7. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont boucler indéfiniment
  - b. vont afficher 11 nombres
  - c. vont afficher 10 nombres
  - d. vont afficher 9 nombres

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(a, b);`
- c. `echange(a++, b++);`
- d. `echange(*a, *b);`

9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100110
- c. 111011010100000
- d. 111011010101000

10. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand `a` est un entier impair
- c. Affiche bonjour quand `a` est un entier pair
- d. N'affiche rien (quelque soit la valeur de `a`)

11. `printf("%c", A)`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. n'a pas de sens
- c. la valeur de `a` tout simplement
- d. l'adresse de la variable `a`

13. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

14. `if (a%2 = 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier impair
- b. Affiche bonjour quand `a` est un entier pair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. N'affiche rien (quelque soit la valeur de `a`)

15. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur FAUX
- b. n'a pas de valeur
- c. a pour valeur VRAI
- d. entraîne l'affichage d'un message d'erreur

16. L'expression `a<=1`
- a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Diminue de 1 la valeur de  $a$
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
17. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. le contenu de la variable
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
20. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

# Sujet n° 55

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 0
2. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101111
  - b. 111011010100110
  - c. 111011010101000
  - d. 111011010100000
4. L'expression `a<=1`
  - a. Réalise une affectation
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Diminue de 1 la valeur de  $a$
  - d. Utilise les opérateurs `<` et `=`
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(*a, *b) ;`
  - b. `echange(a++, b++) ;`
  - c. `echange(&a, &b) ;`
  - d. `echange(a, b) ;`
6. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable

7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 58
  - 62
  - 68
9. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
  - `c` vaut `'a'`
  - `c` vaut `'A'`
  - Un message d'erreur s'affiche
10. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
  - n'a pas de valeur
  - a pour valeur VRAI
11. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
12. `printf("%i", 'A')`
- Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
13. `printf("%c", A)`
- Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`

15. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
  - Un mot-clef du langage C
  - Un identificateur du langage C
16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
17. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 9 nombres
18. `if (a<5) printf("Bonjour"); a=a+1 ;`
- N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
19. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
20. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
 - Affiche bonjour quand `a` est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`

Sujet n° 56

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. Réalise une affectation
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Diminue de 1 la valeur de a
2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. n'a pas de valeur
 - b. a pour valeur FAUX
 - c. a pour valeur VRAI
 - d. entraîne l'affichage d'un message d'erreur
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(*a, *b);`
 - b. `echange(a, b);`
 - c. `echange(a++, b++);`
 - d. `echange(&a, &b);`
6. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`

7. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
 - Affiche bonjour quand a est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
8. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Augmente la valeur de a quelque soit a
9. L'adresse d'une variable c'est :
- le contenu de la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010101111
 - 111011010100000
 - 111011010100110
11. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont afficher 10 nombres
 - vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
12. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
13. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à $t[10]$ la valeur 10
 - Permettent de d'affecter à $t[10]$ la valeur 0
 - Permettent de d'affecter à $t[10]$ la valeur 45
 - Permettent de d'affecter à $t[10]$ la valeur 100
14. Après `char c; c='a'; c=c+1;`
- c vaut 'a'
 - c vaut 'A'
 - c vaut 'b'
 - Un message d'erreur s'affiche

15. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
16. `if` est :
- a. Un mot-clef du langage C
 - b. Un opérateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un identificateur du langage C
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 58
 - b. 54
 - c. 62
 - d. 68
18. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier impair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier pair
19. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de *a* tout simplement
 - b. ce vers quoi pointe le pointeur *a*
 - c. l'adresse de la variable *a*
 - d. n'a pas de sens
20. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*

Sujet n° 57

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100110
- c. 111011010101111
- d. 111011010100000

2. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

3. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

4. `printf("%c", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

5. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 0
- c. Permettent de d'affecter à `t[10]` la valeur 45
- d. Permettent de d'affecter à `t[10]` la valeur 10

6. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur FAUX
- b. entraîne l'affichage d'un message d'erreur
- c. n'a pas de valeur
- d. a pour valeur VRAI

7. `if` est :

- a. Un opérateur du langage C
- b. Un mot-clef du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un identificateur du langage C

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b) ;`
- b. `echange(&a, &b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(a, b) ;`

9. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont afficher 10 nombres
- c. vont afficher 9 nombres
- d. vont boucler indéfiniment

10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. la valeur de `a` tout simplement
- c. n'a pas de sens
- d. l'adresse de la variable `a`

11. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. le numéro de la case mémoire où le contenu de la variable est stocké
- c. le contenu de la variable
- d. ce vers quoi pointe la variable

12. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. la valeur de `a` tout simplement
- c. l'adresse de la variable `a`
- d. n'a pas de sens

13. L'expression `a<=1`

- a. Réalise une affectation
- b. Utilise les opérateurs `<` et `=`
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Diminue de 1 la valeur de `a`

14. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère `'A'`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère `'A'`

15. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 58
- b. 62
- c. 54
- d. 68

16. Après `char c; c='a'; c=c+1;`
- a. `c` vaut `'a'`
 - b. Un message d'erreur s'affiche
 - c. `c` vaut `'A'`
 - d. `c` vaut `'b'`
17. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand `a` est un entier pair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier impair
18. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand `a` est un entier impair
 - b. Affiche bonjour quand `a` est un entier pair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`
20. `if (a<5) printf("Bonjour"); a=a+1;`
- a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - b. Augmente la valeur de `a` quelque soit `a`
 - c. Affiche bonjour quelque soit `a`
 - d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`

Sujet n° 58

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. la valeur de `a` tout simplement
2. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 68
 - c. 54
 - d. 58
3. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère `'A'`
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(*a, *b) ;`
 - b. `echange(a++, b++) ;`
 - c. `echange(&a, &b) ;`
 - d. `echange(a, b) ;`
5. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de `a`
6. L'adresse d'une variable c'est :
 - a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. le contenu de la variable
 - c. l'adresse d'un pointeur sur la variable
 - d. ce vers quoi pointe la variable
7. `if` est :
 - a. Un opérateur du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un mot-clef du langage C
 - d. Un identificateur du langage C

8. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010100000
 - 111011010101111
 - 111011010100110
11. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable *a*
 - ce vers quoi pointe le pointeur *a*
 - la valeur de *a* tout simplement
 - n'a pas de sens
12. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
13. Après `char c ; c='a' ; c=c+1 ;`
- c* vaut 'a'
 - Un message d'erreur s'affiche
 - c* vaut 'b'
 - c* vaut 'A'
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
15. Les instructions
- ```

i=0 ;
while(i<10)
 printf("%i ", i) ;
 i++ ;

```
- vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
  - vont afficher 11 nombres

16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
  - b. entraîne l'affichage d'un message d'erreur
  - c. n'a pas de valeur
  - d. a pour valeur VRAI
17. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de *a*)
  - c. Affiche bonjour quand *a* est un entier pair
  - d. Affiche bonjour quand *a* est un entier impair
18. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Augmente la valeur de *a* quelque soit *a*
  - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - d. Affiche bonjour quelque soit *a*
19. `if (a%2 = 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand *a* est un entier impair
  - c. Affiche bonjour quand *a* est un entier pair
  - d. N'affiche rien (quelque soit la valeur de *a*)
20. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'

# Sujet n° 59

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
  - a. Un identificateur du langage C
  - b. Une commande qu'on tape dans la fenêtre de commande
  - c. Un mot-clef du langage C
  - d. Un opérateur du langage C
2. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
3. `printf("%i", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
4. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur VRAI
  - b. a pour valeur FAUX
  - c. entraîne l'affichage d'un message d'erreur
  - d. n'a pas de valeur
5. Les instructions 

```
i=0 ;
 while(i<10)
 printf("%i ", i) ;
 i++ ;
```

  - a. vont afficher 9 nombres
  - b. vont afficher 11 nombres
  - c. vont boucler indéfiniment
  - d. vont afficher 10 nombres
6. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 68
  - c. 54
  - d. 58

7. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101000
  - 111011010100110
  - 111011010101111
9. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de  $a$
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
11. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
  - l'adresse d'un pointeur sur la variable
  - le contenu de la variable
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
13. `printf("%c", A)`
- Affiche le caractère '`A`'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
14. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair

15. Après `char c ; c='a' ; c=c+1 ;`
- a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'b'`
  - d. `c` vaut `'A'`
16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a++, b++) ;`
  - b. `echange(*a, *b) ;`
  - c. `echange(a, b) ;`
  - d. `echange(&a, &b) ;`
17. `if (a%2 == 0) printf("bonjour") ;`
- a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand `a` est un entier pair
  - d. Affiche bonjour quand `a` est un entier impair
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - b. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
20. `printf("%c", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`

# Sujet n° 60

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.



## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. l'adresse de la variable `a`
  - c. n'a pas de sens
  - d. la valeur de `a` tout simplement
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a++, b++) ;`
  - b. `echange(a, b) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(&a, &b) ;`
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier pair
4. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère '`A`'
  - b. Affiche le caractère '`A`'
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
5. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 0
6. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
  - a. vont afficher 11 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 9 nombres
  - d. vont afficher 10 nombres

7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. la valeur de `a` tout simplement
8. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Diminue de 1 la valeur de `a`
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
10. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur VRAI
  - b. a pour valeur FAUX
  - c. entraîne l'affichage d'un message d'erreur
  - d. n'a pas de valeur
11. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier pair
  - d. Affiche bonjour quand `a` est un entier impair
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101000
  - c. 111011010101111
  - d. 111011010100110
13. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. le contenu de la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. ce vers quoi pointe la variable
14. `printf("%i", A)`
  - a. Affiche le caractère '`A`'
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère '`A`'

15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 54
  - 58
  - 68
16. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
17. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - Augmente la valeur de *a* quelque soit *a*
18. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
19. `if` est :
- Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
  - Un identificateur du langage C
20. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
  - c* vaut 'A'
  - c* vaut 'a'
  - c* vaut 'b'

# Sujet n° 61

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Réalise une affectation
  - c. Diminue de 1 la valeur de  $a$
  - d. Utilise les opérateurs `<` et `=`
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. entraîne l'affichage d'un message d'erreur
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101000
  - c. 111011010101111
  - d. 111011010100110
4. Les instructions `i=0 ;`  

```
while(i<10)
 printf("%i ", i);
 i++ ;
```

  - a. vont boucler indéfiniment
  - b. vont afficher 11 nombres
  - c. vont afficher 9 nombres
  - d. vont afficher 10 nombres
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 62
  - c. 68
  - d. 58
6. Après `char c ; c='a' ; c=c+1 ;`
  - a.  $c$  vaut `'b'`
  - b.  $c$  vaut `'A'`
  - c. Un message d'erreur s'affiche
  - d.  $c$  vaut `'a'`

7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. l'adresse de la variable `a`
  - d. la valeur de `a` tout simplement
8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 0
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - b. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - c. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
10. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
11. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un identificateur du langage C
  - c. Un opérateur du langage C
  - d. Un mot-clef du langage C
12. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
13. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Affiche bonjour quand `a` est un entier pair
14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. ce vers quoi pointe le pointeur `a`



15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(&a, &b) ;`
- d. `echange(*a, *b) ;`

16. L'adresse d'une variable c'est :

- a. le contenu de la variable
- b. l'adresse d'un pointeur sur la variable
- c. le numéro de la case mémoire où le contenu de la variable est stocké
- d. ce vers quoi pointe la variable

17. `printf("%i", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

18. `if (a%2 == 0) printf("bonjour") ;`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand `a` est un entier impair
- c. N'affiche rien (quelque soit la valeur de `a`)
- d. Affiche bonjour quand `a` est un entier pair

19. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Augmente la valeur de `a` quelque soit `a`
- b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- c. Affiche bonjour quelque soit `a`
- d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`

20. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère 'A'

# Sujet n° 62

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le code ASCII du caractère 'A'
2. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Affiche bonjour quelque soit  $a$
  - b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - d. Augmente la valeur de  $a$  quelque soit  $a$
3. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Utilise les opérateurs `<` et `=`
  - c. Réalise une affectation
  - d. Diminue de 1 la valeur de  $a$
4. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
5. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable A
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
6. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
7. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur FAUX
  - c. n'a pas de valeur
  - d. a pour valeur VRAI

8. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont afficher 11 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
  - vont afficher 9 nombres
9. `printf("%c", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
10. `if` est :
- Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
  - Un mot-clef du langage C
11. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand *a* est un entier impair
  - Affiche bonjour quand *a* est un entier pair
  - N'affiche rien (quelque soit la valeur de *a*)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
12. On suppose que **a** a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur **a**
  - la valeur de **a** tout simplement
  - l'adresse de la variable **a**
  - n'a pas de sens
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
  - 58
  - 54
  - 62
14. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand *a* est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier impair
15. Après `char c ; c='a' ; c=c+1 ;`
- c* vaut 'A'
  - c* vaut 'b'
  - Un message d'erreur s'affiche
  - c* vaut 'a'

16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
17. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b);`
  - `echange(a, b);`
  - `echange(&a, &b);`
  - `echange(a++, b++);`
20. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100000
  - 111011010101000
  - 111011010100110

# Sujet n° 63

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Affiche bonjour quand  $a$  est un entier impair
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101111
  - c. 111011010100110
  - d. 111011010101000
3. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère 'A'
4. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un mot-clef du langage C
  - c. Un opérateur du langage C
  - d. Un identificateur du langage C
5. `if (a%2 = 0) printf("bonjour");`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
6. On suppose que  $a$  a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. l'adresse de la variable  $a$
  - b. n'a pas de sens
  - c. la valeur de  $a$  tout simplement
  - d. ce vers quoi pointe le pointeur  $a$

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(&a, &b) ;`
- c. `echange(*a, *b) ;`
- d. `echange(a, b) ;`

8. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- b. Augmente la valeur de `a` quelque soit `a`
- c. Affiche bonjour quelque soit `a`
- d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`

9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
- b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
- c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
- d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`

10. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

11. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Diminue de 1 la valeur de `a`
- d. Réalise une affectation

12. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 100

13. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```

- a. vont afficher 10 nombres
- b. vont afficher 9 nombres
- c. vont boucler indéfiniment
- d. vont afficher 11 nombres

14. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 54
- b. 68
- c. 62
- d. 58

15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
  - b. l'adresse de la variable `a`
  - c. ce vers quoi pointe le pointeur `a`
  - d. n'a pas de sens
16. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
17. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
18. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'b'`
  - b. `c` vaut `'a'`
  - c. `c` vaut `'A'`
  - d. Un message d'erreur s'affiche
19. L'adresse d'une variable c'est :
- a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. l'adresse d'un pointeur sur la variable
  - c. ce vers quoi pointe la variable
  - d. le contenu de la variable
20. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
  - b. a pour valeur VRAI
  - c. entraîne l'affichage d'un message d'erreur
  - d. n'a pas de valeur

# Sujet n° 64

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
  - a. Diminue de 1 la valeur de  $a$
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
2. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand  $a$  est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Affiche bonjour quand  $a$  est un entier impair
3. `if` est :
  - a. Un opérateur du langage C
  - b. Un mot-clef du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un identificateur du langage C
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(*a, *b);`
  - b. `echange(&a, &b);`
  - c. `echange(a++, b++);`
  - d. `echange(a, b);`
5. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :
  - a. `111011010100000`
  - b. `111011010101000`
  - c. `111011010100110`
  - d. `111011010101111`
6. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`

7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
9. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut 'A'
  - Un message d'erreur s'affiche
  - `c` vaut 'a'
  - `c` vaut 'b'
10. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur
  - n'a pas de valeur
  - a pour valeur VRAI
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
12. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le code ASCII du caractère stocké dans la variable A
  - Affiche le caractère 'A'
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
14. L'adresse d'une variable c'est :
- le contenu de la variable
  - l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable

15. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
16. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
  - Affiche bonjour quand *a* est un entier pair
  - N'affiche rien (quelque soit la valeur de *a*)
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
  - 54
  - 58
  - 62
18. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - Augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour quelque soit *a*
19. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
- vont boucler indéfiniment
  - vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont afficher 10 nombres
20. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'



# Sujet n° 65

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101000
  - b. 111011010100110
  - c. 111011010101111
  - d. 111011010100000
2. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 45
3. L'expression `a<=1`
  - a. Réalise une affectation
  - b. Diminue de 1 la valeur de `a`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Utilise les opérateurs `<` et `=`
4. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 68
  - c. 54
  - d. 58
6. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un mot-clef du langage C
  - c. Un identificateur du langage C
  - d. Un opérateur du langage C
7. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. le contenu de la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable

8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

9. `if (a<5) printf("Bonjour"); a=a+1;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
- b. Augmente la valeur de *a* quelque soit *a*
- c. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
- d. Affiche bonjour quelque soit *a*

10. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable *A*
- b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- c. Affiche le code ASCII du caractère '*A*'
- d. Affiche le caractère '*A*'

11. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand *a* est un entier impair
- c. N'affiche rien (quelque soit la valeur de *a*)
- d. Affiche bonjour quand *a* est un entier pair

12. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable *A*
- b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- c. Affiche le code ASCII du caractère '*A*'
- d. Affiche le caractère '*A*'

13. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur *a*
- b. n'a pas de sens
- c. l'adresse de la variable *a*
- d. la valeur de *a* tout simplement

14. Les instructions `i=0;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 9 nombres

15. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le caractère '*A*'
- c. Affiche le code ASCII du caractère '*A*'
- d. Affiche le code ASCII du caractère stocké dans la variable *A*

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. ce vers quoi pointe le pointeur `a`
  - c. n'a pas de sens
  - d. la valeur de `a` tout simplement
17. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'A'`
  - b. `c` vaut `'b'`
  - c. `c` vaut `'a'`
  - d. Un message d'erreur s'affiche
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
  - b. a pour valeur VRAI
  - c. entraîne l'affichage d'un message d'erreur
  - d. n'a pas de valeur
19. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand `a` est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand `a` est un entier pair
  - d. N'affiche rien (quelque soit la valeur de `a`)
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a++, b++) ;`
  - b. `echange(&a, &b) ;`
  - c. `echange(a, b) ;`
  - d. `echange(*a, *b) ;`

# Sujet n° 66

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
2. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
  - a. vont afficher 10 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 9 nombres
  - d. vont afficher 11 nombres
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier pair
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de `a`)
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a++, b++) ;`
  - b. `echange(a, b) ;`
  - c. `echange(&a, &b) ;`
  - d. `echange(*a, *b) ;`
5. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. ce vers quoi pointe le pointeur `a`
6. L'expression `a<=1`
  - a. Utilise les opérateurs `<` et `=`
  - b. Réalise une affectation
  - c. Diminue de 1 la valeur de `a`
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon



7. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
8. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère 'A'
9. `if` est :
- Un opérateur du langage C
  - Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un mot-clef du langage C
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
  - 111011010100110
  - 111011010100000
  - 111011010101111
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
13. `printf("%c", A)`
- Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
  - b. 58
  - c. 68
  - d. 54
16. L'adresse d'une variable c'est :
- a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
17. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - c. Affiche le code ASCII du caractère stocké dans la variable A
  - d. Affiche le caractère 'A'
18. Après `char c ; c='a' ; c=c+1 ;`
- a. c vaut 'b'
  - b. c vaut 'A'
  - c. c vaut 'a'
  - d. Un message d'erreur s'affiche
19. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le caractère 'A'
20. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur VRAI
  - c. n'a pas de valeur
  - d. a pour valeur FAUX

# Sujet n° 67

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'b'`
  - b. Un message d'erreur s'affiche
  - c. `c` vaut `'a'`
  - d. `c` vaut `'A'`
2. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 62
  - c. 58
  - d. 54
4. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Diminue de 1 la valeur de `a`
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
5. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 100
6. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. l'adresse d'un pointeur sur la variable
  - d. le contenu de la variable
7. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010100110
  - c. 111011010101111
  - d. 111011010101000

8. `if` est :
- Un mot-clef du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur VRAI
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
10. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier impair
  - Affiche bonjour quand  $a$  est un entier pair
11. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère 'A'
12. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
13. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
14. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b);`
  - `echange(&a, &b);`
  - `echange(a, b);`
  - `echange(a++, b++);`

16. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont afficher 10 nombres  
b. vont boucler indéfiniment  
c. vont afficher 11 nombres  
d. vont afficher 9 nombres
17. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :  
a. n'a pas de sens  
b. la valeur de `a` tout simplement  
c. ce vers quoi pointe le pointeur `a`  
d. l'adresse de la variable `a`
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :  
a. la valeur de `a` tout simplement  
b. ce vers quoi pointe le pointeur `a`  
c. n'a pas de sens  
d. l'adresse de la variable `a`
19. `printf("%c", 'A')`  
a. Affiche le caractère 'A'  
b. Affiche le code ASCII du caractère 'A'  
c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`  
d. Affiche le code ASCII du caractère stocké dans la variable `A`
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?  
a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`  
b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`  
c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`  
d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

# Sujet n° 68

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.



## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 10 nombres
- b. vont afficher 9 nombres
- c. vont afficher 11 nombres
- d. vont boucler indéfiniment

2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(*a, *b) ;`
- c. `echange(a, b) ;`
- d. `echange(&a, &b) ;`

3. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur VRAI
- b. entraîne l'affichage d'un message d'erreur
- c. a pour valeur FAUX
- d. n'a pas de valeur

4. L'expression `a<=1`

- a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- b. Utilise les opérateurs `<` et `=`
- c. Diminue de 1 la valeur de  $a$
- d. Réalise une affectation

5. Après `char c ; c='a' ; c=c+1 ;`

- a. Un message d'erreur s'affiche
- b.  $c$  vaut `'b'`
- c.  $c$  vaut `'a'`
- d.  $c$  vaut `'A'`

6. `if` est :

- a. Un identificateur du langage C
- b. Un mot-clef du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un opérateur du langage C

7. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Déclenche le message d'erreur `invalid lvalue in assignment`
8. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère 'A'
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 68
  - 54
  - 58
11. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de  $a$ )
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - Affiche bonjour quand  $a$  est un entier pair
12. On suppose que  $a$  a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable  $a$
  - la valeur de  $a$  tout simplement
  - ce vers quoi pointe le pointeur  $a$
  - n'a pas de sens
13. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère 'A'
14. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à  $t[10]$  la valeur 45
  - Permettent de d'affecter à  $t[10]$  la valeur 0
  - Permettent de d'affecter à  $t[10]$  la valeur 100
  - Permettent de d'affecter à  $t[10]$  la valeur 10

15. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100110
  - 111011010101000
  - 111011010100000
16. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
17. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
18. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
19. L'adresse d'une variable c'est :
- le contenu de la variable
  - l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens

# Sujet n° 69

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 62
  - c. 68
  - d. 54
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. a pour valeur FAUX
3. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - c. Affiche le code ASCII du caractère stocké dans la variable A
  - d. Affiche le caractère 'A'
4. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de *a*)
  - c. Affiche bonjour quand *a* est un entier impair
  - d. Affiche bonjour quand *a* est un entier pair
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
6. `if (a%2 = 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de *a*)
  - c. Affiche bonjour quand *a* est un entier pair
  - d. Affiche bonjour quand *a* est un entier impair
7. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable A
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable A

8. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
9. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'A'`
  - `c` vaut `'a'`
  - `c` vaut `'b'`
  - Un message d'erreur s'affiche
10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - n'a pas de sens
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
  - `echange(&a, &b) ;`
  - `echange(a, b) ;`
  - `echange(*a, *b) ;`
13. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
  - Réalise une affectation
  - Diminue de 1 la valeur de `a`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
14. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
  - l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le contenu de la variable
15. Les instructions
- ```

i=0 ;
while(i<10)
    printf("%i ", i) ;
    i++ ;

```
- vont afficher 9 nombres
 - vont afficher 10 nombres
 - vont afficher 11 nombres
 - vont boucler indéfiniment

16. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010100110
18. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 100
19. `if` est :
- a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Un identificateur du langage C
20. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Augmente la valeur de *a* quelque soit *a*
 - b. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - c. Affiche bonjour quelque soit *a*
 - d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*

Sujet n° 70

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
2. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'A'`
 - b. `c` vaut `'a'`
 - c. Un message d'erreur s'affiche
 - d. `c` vaut `'b'`
3. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand `a` est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand `a` est un entier pair
 - d. N'affiche rien (quelque soit la valeur de `a`)
4. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère `'A'`
5. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour quelque soit `a`
 - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - d. Augmente la valeur de `a` quelque soit `a`
6. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`
7. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b) ;`
 - b. `echange(a++, b++) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(&a, &b) ;`
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 68
 - b. 62
 - c. 58
 - d. 54
10. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. entraîne l'affichage d'un message d'erreur
11. L'expression `a<=1`
- a. Réalise une affectation
 - b. Diminue de 1 la valeur de `a`
 - c. Utilise les opérateurs `<` et `=`
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
12. L'adresse d'une variable c'est :
- a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
 - b. ce vers quoi pointe le pointeur `a`
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
14. `printf("%c", A)`
- a. Affiche le caractère '`A`'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère '`A`'
15. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand `a` est un entier pair
 - b. Affiche bonjour quand `a` est un entier impair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`

16. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont afficher 9 nombres
b. vont boucler indéfiniment
c. vont afficher 11 nombres
d. vont afficher 10 nombres
17. `if` est :
a. Un identificateur du langage C
b. Une commande qu'on tape dans la fenêtre de commande
c. Un mot-clef du langage C
d. Un opérateur du langage C
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
a. 111011010100110
b. 111011010101111
c. 111011010101000
d. 111011010100000
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
a. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
c. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
d. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
20. `printf("%i", 'A')`
a. Affiche le code ASCII du caractère stocké dans la variable `A`
b. Affiche le caractère `'A'`
c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
d. Affiche le code ASCII du caractère `'A'`

Sujet n° 71

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand a est un entier pair
 - b. Affiche bonjour quand a est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. N'affiche rien (quelque soit la valeur de a)
2. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 11 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 9 nombres
 - d. vont afficher 10 nombres
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 68
 - c. 54
 - d. 62
4. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Affiche bonjour quelque soit a
 - c. Augmente la valeur de a quelque soit a
 - d. Affiche bonjour et augmente la valeur de a quelque soit a
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
6. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un mot-clef du langage C
 - c. Un opérateur du langage C
 - d. Un identificateur du langage C

7. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
8. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. l'adresse d'un pointeur sur la variable
 - d. le contenu de la variable
9. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
10. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
11. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur FAUX
 - c. n'a pas de valeur
 - d. a pour valeur VRAI
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 45
13. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand *a* est un entier pair
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand *a* est un entier impair
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - la valeur de `a` tout simplement
16. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010101111
 - 111011010101000
 - 111011010100110
18. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
 - `echange(*a, *b) ;`
20. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
 - `c` vaut `'b'`
 - Un message d'erreur s'affiche
 - `c` vaut `'A'`

Sujet n° 72

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier pair
 - d. Affiche bonjour quand a est un entier impair
2. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
3. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à $t[10]$ la valeur 45
 - b. Permettent de d'affecter à $t[10]$ la valeur 100
 - c. Permettent de d'affecter à $t[10]$ la valeur 0
 - d. Permettent de d'affecter à $t[10]$ la valeur 10
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010101000
 - c. 111011010101111
 - d. 111011010100000
5. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Augmente la valeur de a quelque soit a
 - d. Affiche bonjour et augmente la valeur de a quelque soit a
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables a et b on doit écrire :
 - a. `echange(&a, &b);`
 - b. `echange(*a, *b);`
 - c. `echange(a, b);`
 - d. `echange(a++, b++);`

7. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

8. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 9 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 11 nombres

9. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le caractère 'A'

10. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

11. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Utilise les opérateurs `<` et `=`
- c. Diminue de 1 la valeur de *a*
- d. Réalise une affectation

12. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un mot-clef du langage C
- c. Un identificateur du langage C
- d. Un opérateur du langage C

13. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. la valeur de *a* tout simplement
- b. ce vers quoi pointe le pointeur *a*
- c. n'a pas de sens
- d. l'adresse de la variable *a*

14. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de *a*)
- c. Affiche bonjour quand *a* est un entier pair
- d. Affiche bonjour quand *a* est un entier impair

15. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
16. Après `char c ; c='a' ; c=c+1 ;`
- a. Un message d'erreur s'affiche
 - b. c vaut 'A'
 - c. c vaut 'a'
 - d. c vaut 'b'
17. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. n'a pas de valeur
 - b. a pour valeur VRAI
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur FAUX
18. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
 - b. 58
 - c. 68
 - d. 54
19. L'adresse d'une variable c'est :
- a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. n'a pas de sens
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`

Sujet n° 73

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
2. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 100
4. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010101000
6. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de *a* tout simplement
 - b. ce vers quoi pointe le pointeur *a*
 - c. n'a pas de sens
 - d. l'adresse de la variable *a*
7. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand *a* est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier pair

8. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. le contenu de la variable
 - d. ce vers quoi pointe la variable
9. `if` est :
 - a. Un mot-clef du langage C
 - b. Un opérateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un identificateur du langage C
10. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. entraîne l'affichage d'un message d'erreur
11. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Augmente la valeur de a quelque soit a
 - b. Affiche bonjour et augmente la valeur de a quelque soit a
 - c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 54
 - c. 62
 - d. 58
14. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. ce vers quoi pointe le pointeur `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens

16. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier pair
 - c. Affiche bonjour quand a est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a++, b++) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(a, b) ;`
18. L'expression `a<=1`
- a. Réalise une affectation
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Diminue de 1 la valeur de a
 - d. Utilise les opérateurs `<` et `=`
19. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'b'`
 - b. `c` vaut `'A'`
 - c. Un message d'erreur s'affiche
 - d. `c` vaut `'a'`
20. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- a. vont boucler indéfiniment
  - b. vont afficher 11 nombres
  - c. vont afficher 10 nombres
  - d. vont afficher 9 nombres

# Sujet n° 74

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010100110
- c. 111011010101111
- d. 111011010101000

2. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 58
- b. 54
- c. 68
- d. 62

3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a, b) ;`
- c. `echange(*a, *b) ;`
- d. `echange(a++, b++) ;`

4. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur FAUX
- c. a pour valeur VRAI
- d. n'a pas de valeur

5. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un identificateur du langage C
- c. Un mot-clef du langage C
- d. Un opérateur du langage C

6. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour quelque soit  $a$
- b. Augmente la valeur de  $a$  quelque soit  $a$
- c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
- d. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$



7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. l'adresse de la variable `a`
  - c. n'a pas de sens
  - d. la valeur de `a` tout simplement
8. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le contenu de la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
9. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'A'`
  - b. `c` vaut `'a'`
  - c. `c` vaut `'b'`
  - d. Un message d'erreur s'affiche
10. `printf("%i", 'A')`
  - a. Affiche le caractère `'A'`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
11. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Affiche bonjour quand `a` est un entier pair
  - d. N'affiche rien (quelque soit la valeur de `a`)
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. la valeur de `a` tout simplement
  - c. n'a pas de sens
  - d. ce vers quoi pointe le pointeur `a`
13. L'expression `a--`
  - a. Diminue de 1 la valeur de `a`
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Réalise une affectation
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

15. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
16. `printf("%i", A)`
- Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
17. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
18. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à  $t[10]$  la valeur 100
  - Permettent de d'affecter à  $t[10]$  la valeur 0
  - Permettent de d'affecter à  $t[10]$  la valeur 45
  - Permettent de d'affecter à  $t[10]$  la valeur 10
19. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
20. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'

# Sujet n° 75

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier pair
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
2. `printf("%c", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
3. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. le contenu de la variable
4. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Affiche bonjour quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Augmente la valeur de  $a$  quelque soit  $a$
5. L'expression `a<=1`
  - a. Réalise une affectation
  - b. Utilise les opérateurs `<` et `=`
  - c. Diminue de 1 la valeur de  $a$
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
6. Après `char c ; c='a' ; c=c+1 ;`
  - a.  $c$  vaut 'b'
  - b. Un message d'erreur s'affiche
  - c.  $c$  vaut 'A'
  - d.  $c$  vaut 'a'
7. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à  $t[10]$  la valeur 100
  - b. Permettent de d'affecter à  $t[10]$  la valeur 0
  - c. Permettent de d'affecter à  $t[10]$  la valeur 45
  - d. Permettent de d'affecter à  $t[10]$  la valeur 10

8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 58
  - 68
9. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
10. Les instructions
- ```
i=0 ;
while(i<10)
    printf("%i ", i) ;
    i++ ;
```
- vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 9 nombres
11. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a, b) ;`
 - `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(*a, *b) ;`
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010100000
 - 111011010100110
 - 111011010101000
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - a pour valeur VRAI
 - a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
15. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
 - la valeur de *a* tout simplement
 - l'adresse de la variable *a*
 - n'a pas de sens

16. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
18. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
 - n'a pas de sens
 - la valeur de *a* tout simplement
 - l'adresse de la variable *a*
19. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
20. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C

Sujet n° 76

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010101111
 - c. 111011010101000
 - d. 111011010100110
3. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
4. Après `char c ; c='a' ; c=c+1 ;`
 - a. Un message d'erreur s'affiche
 - b. *c* vaut 'A'
 - c. *c* vaut 'a'
 - d. *c* vaut 'b'
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - b. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
6. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le contenu de la variable
 - c. l'adresse d'un pointeur sur la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
7. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Augmente la valeur de *a* quelque soit *a*
 - b. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - c. Affiche bonjour quelque soit *a*
 - d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(*a, *b) ;`
- c. `echange(a, b) ;`
- d. `echange(a++, b++) ;`

9. L'expression `a<=1`

- a. Diminue de 1 la valeur de a
- b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- c. Réalise une affectation
- d. Utilise les opérateurs `<` et `=`

10. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur FAUX
- c. a pour valeur VRAI
- d. n'a pas de valeur

11. `if` est :

- a. Un mot-clef du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un identificateur du langage C
- d. Un opérateur du langage C

12. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. la valeur de `a` tout simplement
- c. l'adresse de la variable `a`
- d. n'a pas de sens

13. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand a est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand a est un entier pair
- d. N'affiche rien (quelque soit la valeur de a)

14. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

- a. vont afficher 10 nombres
- b. vont boucler indéfiniment
- c. vont afficher 11 nombres
- d. vont afficher 9 nombres

15. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 62
- b. 68
- c. 54
- d. 58

16. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
17. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère 'A'
18. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
19. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - n'a pas de sens
 - la valeur de `a` tout simplement
20. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Déclenche le message d'erreur `invalid lvalue in assignment`

Sujet n° 77

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101111
- c. 111011010101000
- d. 111011010100000

2. `printf("%c", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable A
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le code ASCII du caractère 'A'

3. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 58
- c. 62
- d. 54

4. `if (a<5) printf("Bonjour"); a=a+1;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- b. Affiche bonjour quelque soit a
- c. Affiche bonjour et augmente la valeur de a quelque soit a
- d. Augmente la valeur de a quelque soit a

5. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de a)
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand a est un entier impair
- d. Affiche bonjour quand a est un entier pair

6. L'adresse d'une variable c'est :

- a. le contenu de la variable
- b. ce vers quoi pointe la variable
- c. l'adresse d'un pointeur sur la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

7. Après `char c; c='a'; c=c+1;`

- a. c vaut 'a'
- b. Un message d'erreur s'affiche
- c. c vaut 'b'
- d. c vaut 'A'

8. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - n'a pas de valeur
9. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier pair
 - N'affiche rien (quelque soit la valeur de *a*)
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
11. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
12. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Diminue de 1 la valeur de *a*
 - Réalise une affectation
13. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a++, b++);`
 - `echange(*a, *b);`
 - `echange(a, b);`
 - `echange(&a, &b);`
15. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un mot-clef du langage C

16. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
17. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. ce vers quoi pointe le pointeur *a*
 - b. n'a pas de sens
 - c. la valeur de *a* tout simplement
 - d. l'adresse de la variable *a*
18. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
 - b. la valeur de *a* tout simplement
 - c. ce vers quoi pointe le pointeur *a*
 - d. l'adresse de la variable *a*
19. `printf("%c", A)`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
20. Les instructions
- ```
i=0 ;
 while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- a. vont afficher 10 nombres
  - b. vont afficher 9 nombres
  - c. vont afficher 11 nombres
  - d. vont boucler indéfiniment

# Sujet n° 78

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101111
  - c. 111011010101000
  - d. 111011010100110
3. `printf("%i", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
4. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
5. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Augmente la valeur de *a* quelque soit *a*
  - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - d. Affiche bonjour quelque soit *a*
6. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
7. `printf("%c", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
9. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
10. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
  - Utilise les opérateurs `<` et `=`
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- `a` pour valeur VRAI
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
  - `a` pour valeur FAUX
12. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de `a`)
13. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
14. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
- vont boucler indéfiniment
  - vont afficher 10 nombres
  - vont afficher 11 nombres
  - vont afficher 9 nombres
15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 62
  - 54
  - 68

16. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'A'`
  - b. `c` vaut `'b'`
  - c. `c` vaut `'a'`
  - d. Un message d'erreur s'affiche
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b) ;`
  - b. `echange(&a, &b) ;`
  - c. `echange(a++, b++) ;`
  - d. `echange(*a, *b) ;`
18. `if (a%2 == 0) printf("bonjour") ;`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier pair
  - d. Affiche bonjour quand `a` est un entier impair
19. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
  - b. la valeur de `a` tout simplement
  - c. l'adresse de la variable `a`
  - d. n'a pas de sens
20. `if` est :
- a. Un mot-clef du langage C
  - b. Un opérateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un identificateur du langage C

# Sujet n° 79

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut 'A'
  - b. `c` vaut 'a'
  - c. `c` vaut 'b'
  - d. Un message d'erreur s'affiche
2. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère 'A'
3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 100
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 10
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. l'adresse de la variable `a`
  - d. ce vers quoi pointe le pointeur `a`
5. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère 'A'
6. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 58
  - c. 54
  - d. 68
7. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Affiche bonjour quelque soit `a`
  - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - d. Augmente la valeur de `a` quelque soit `a`

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b);`
- b. `echange(a, b);`
- c. `echange(&a, &b);`
- d. `echange(a++, b++);`

9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echage(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echage(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echage(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echage(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101111
- c. 111011010100110
- d. 111011010101000

11. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère `'A'`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

12. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur FAUX
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur VRAI

13. `if` est :

- a. Un identificateur du langage C
- b. Un mot-clef du langage C
- c. Un opérateur du langage C
- d. Une commande qu'on tape dans la fenêtre de commande

14. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier impair
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier pair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

15. L'adresse d'une variable c'est :

- a. le numéro de la case mémoire où le contenu de la variable est stocké
- b. l'adresse d'un pointeur sur la variable
- c. le contenu de la variable
- d. ce vers quoi pointe la variable

16. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- a. vont afficher 11 nombres
  - b. vont afficher 9 nombres
  - c. vont boucler indéfiniment
  - d. vont afficher 10 nombres
17. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. la valeur de `a` tout simplement
18. L'expression `a<=1`
- a. Réalise une affectation
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Diminue de 1 la valeur de `a`
19. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
20. `if (a%2 == 0) printf("bonjour") ;`
- a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier pair

# Sujet n° 80

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. a pour valeur FAUX
3. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le caractère 'A'
4. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 68
  - c. 58
  - d. 62
6. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010100110
  - c. 111011010101111
  - d. 111011010101000
7. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable

8. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un mot-clef du langage C
  - Un opérateur du langage C
  - Un identificateur du langage C
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
  - `echange(a++, b++) ;`
  - `echange(&a, &b) ;`
  - `echange(*a, *b) ;`
10. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
11. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de `a`
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
12. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
13. Les instructions `i=0 ;`
- ```

while(i<10)
    printf("%i ", i) ;
    i++ ;

```
- vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
14. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45

16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. n'a pas de sens
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`
17. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
18. Après `char c; c='a'; c=c+1;`
- a. `c` vaut `'b'`
 - b. Un message d'erreur s'affiche
 - c. `c` vaut `'a'`
 - d. `c` vaut `'A'`
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens

Sujet n° 81

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
 - a. Un mot-clef du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010101000
 - c. 111011010100110
 - d. 111011010100000
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 54
 - b. 58
 - c. 62
 - d. 68
4. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
5. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère 'A'
6. `printf("%c", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
7. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de *a* tout simplement
 - b. n'a pas de sens
 - c. ce vers quoi pointe le pointeur *a*
 - d. l'adresse de la variable *a*

8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

9. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 100
- c. Permettent de d'affecter à `t[10]` la valeur 45
- d. Permettent de d'affecter à `t[10]` la valeur 10

10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++);`
- b. `echange(*a, *b);`
- c. `echange(a, b);`
- d. `echange(&a, &b);`

11. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur FAUX
- b. entraîne l'affichage d'un message d'erreur
- c. a pour valeur VRAI
- d. n'a pas de valeur

12. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de `a`)
- b. Affiche bonjour quand `a` est un entier pair
- c. Affiche bonjour quand `a` est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

13. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Utilise les opérateurs `<` et `=`
- c. Réalise une affectation
- d. Diminue de 1 la valeur de `a`

14. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 9 nombres

15. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour quelque soit a
 - b. Affiche bonjour et augmente la valeur de a quelque soit a
 - c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - d. Augmente la valeur de a quelque soit a
16. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
 - b. le contenu de la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
17. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
18. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
 - b. c vaut 'a'
 - c. c vaut 'b'
 - d. c vaut 'A'
19. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier impair
 - c. N'affiche rien (quelque soit la valeur de a)
 - d. Affiche bonjour quand a est un entier pair
20. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable a
 - b. ce vers quoi pointe le pointeur a
 - c. n'a pas de sens
 - d. la valeur de a tout simplement

Sujet n° 82

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère 'A'
2. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour et augmente la valeur de a quelque soit a
 - b. Affiche bonjour quelque soit a
 - c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - d. Augmente la valeur de a quelque soit a
3. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand a est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier pair
 - d. N'affiche rien (quelque soit la valeur de a)
4. `if` est :
 - a. Un mot-clef du langage C
 - b. Un opérateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un identificateur du langage C
5. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
6. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. entraîne l'affichage d'un message d'erreur
 - c. n'a pas de valeur
 - d. a pour valeur VRAI
7. L'expression `a<=1`
 - a. Diminue de 1 la valeur de a
 - b. Réalise une affectation
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Utilise les opérateurs `<` et `=`

8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010100000
 - 111011010100110
 - 111011010101000
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 54
 - 68
 - 58
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
12. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
13. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de *a* tout simplement
 - l'adresse de la variable *a*
 - ce vers quoi pointe le pointeur *a*
 - n'a pas de sens
14. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - la valeur de *a* tout simplement
 - ce vers quoi pointe le pointeur *a*
 - l'adresse de la variable *a*
15. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable

16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(*a, *b) ;`
- d. `echange(&a, &b) ;`

17. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le code ASCII du caractère 'A'

18. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut 'b'
- b. Un message d'erreur s'affiche
- c. `c` vaut 'A'
- d. `c` vaut 'a'

19. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

- a. vont afficher 11 nombres
- b. vont afficher 9 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment

20. `printf("%i", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère 'A'

Sujet n° 83

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. Réalise une affectation
 - c. Diminue de 1 la valeur de a
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(&a, &b) ;`
 - b. `echange(a, b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(a++, b++) ;`
3. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand a est un entier impair
 - b. Affiche bonjour quand a est un entier pair
 - c. N'affiche rien (quelque soit la valeur de a)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
4. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 10 nombres
 - b. vont afficher 11 nombres
 - c. vont boucler indéfiniment
 - d. vont afficher 9 nombres
5. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'b'`
 - b. `c` vaut `'A'`
 - c. Un message d'erreur s'affiche
 - d. `c` vaut `'a'`
6. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Affiche bonjour et augmente la valeur de a quelque soit a
 - c. Augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a

7. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur VRAI
 - d. a pour valeur FAUX
8. L'adresse d'une variable c'est :
 - a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. ce vers quoi pointe la variable
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. la valeur de `a` tout simplement
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. la valeur de `a` tout simplement
11. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 100
 - d. Permettent de d'affecter à `t[10]` la valeur 10
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 68
 - c. 54
 - d. 62
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010100110
 - c. 111011010101000
 - d. 111011010100000
14. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère '`A`'
 - b. Affiche le caractère '`A`'
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

15. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
17. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère `'A'`
18. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
19. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
20. `if` est :
- Un opérateur du langage C
 - Un mot-clef du langage C
 - Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande

Sujet n° 84

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
2. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - c. Augmente la valeur de *a* quelque soit *a*
 - d. Affiche bonjour quelque soit *a*
3. On suppose que **a** a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable **a**
 - b. ce vers quoi pointe le pointeur **a**
 - c. la valeur de **a** tout simplement
 - d. n'a pas de sens
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
5. `printf("%c", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
6. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. a pour valeur VRAI

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b);`
- b. `echange(a++, b++);`
- c. `echange(*a, *b);`
- d. `echange(&a, &b);`

8. Après `char c; c='a'; c=c+1;`

- a. `c` vaut `'a'`
- b. `c` vaut `'A'`
- c. `c` vaut `'b'`
- d. Un message d'erreur s'affiche

9. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. le numéro de la case mémoire où le contenu de la variable est stocké
- c. ce vers quoi pointe la variable
- d. le contenu de la variable

10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

11. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand `a` est un entier pair
- c. Affiche bonjour quand `a` est un entier impair
- d. N'affiche rien (quelque soit la valeur de `a`)

12. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Diminue de 1 la valeur de `a`
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Réalise une affectation

13. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier pair
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable `a`
- b. n'a pas de sens
- c. ce vers quoi pointe le pointeur `a`
- d. la valeur de `a` tout simplement

15. `if` est :
- Un opérateur du langage C
 - Un identificateur du langage C
 - Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
16. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
17. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
18. `printf("%c", A)`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 54
 - 62
 - 68
20. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 10 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres

Sujet n° 85

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
2. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - c. Affiche bonjour quelque soit *a*
 - d. Augmente la valeur de *a* quelque soit *a*
3. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
 - a. vont afficher 11 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 9 nombres
 - d. vont afficher 10 nombres
4. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand a est un entier pair
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier impair
5. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
6. Après `char c; c='a'; c=c+1;`
 - a. Un message d'erreur s'affiche
 - b. *c* vaut 'b'
 - c. *c* vaut 'A'
 - d. *c* vaut 'a'

7. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
8. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
9. L'expression `a--`
- Diminue de 1 la valeur de a
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010100110
 - 111011010101111
 - 111011010101000
11. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
12. `printf("%c", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
13. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable a
 - la valeur de a tout simplement
 - ce vers quoi pointe le pointeur a
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
 - n'a pas de valeur
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur FAUX

15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
 - b. 68
 - c. 54
 - d. 58
16. `if` est :
- a. Un opérateur du langage C
 - b. Un identificateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un mot-clef du langage C
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. la valeur de `a` tout simplement
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
19. `printf("%i", A)`
- a. Affiche le caractère '`A`'
 - b. Affiche le code ASCII du caractère '`A`'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(*a, *b);`
 - b. `echange(&a, &b);`
 - c. `echange(a++, b++);`
 - d. `echange(a, b);`

Sujet n° 86

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010101000
- c. 111011010100110
- d. 111011010100000

2. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand a est un entier pair
- c. N'affiche rien (quelque soit la valeur de a)
- d. Affiche bonjour quand a est un entier impair

3. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Réalise une affectation
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Diminue de 1 la valeur de a

4. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit a
- b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- c. Augmente la valeur de a quelque soit a
- d. Affiche bonjour et augmente la valeur de a quelque soit a

5. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. n'a pas de sens
- c. l'adresse de la variable `a`
- d. ce vers quoi pointe le pointeur `a`

6. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur FAUX
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur VRAI

7. `printf("%c", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a++, b++) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(a, b) ;`
9. `printf("%c", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
10. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand `a` est un entier pair
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand `a` est un entier impair
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
12. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut 'b'
 - b. `c` vaut 'a'
 - c. `c` vaut 'A'
 - d. Un message d'erreur s'affiche
13. `if` est :
- a. Un opérateur du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un mot-clef du langage C
 - d. Un identificateur du langage C
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 68
 - b. 62
 - c. 58
 - d. 54
15. L'adresse d'une variable c'est :
- a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. l'adresse d'un pointeur sur la variable
 - c. ce vers quoi pointe la variable
 - d. le contenu de la variable

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

17. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 100

18. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère `'A'`

19. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 11 nombres
- c. vont afficher 9 nombres
- d. vont afficher 10 nombres

20. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère `'A'`

Sujet n° 87

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
2. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. l'adresse d'un pointeur sur la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
3. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
4. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100
5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010100110
 - c. 111011010101000
 - d. 111011010100000
6. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
 - a. vont afficher 11 nombres
 - b. vont afficher 9 nombres
 - c. vont boucler indéfiniment
 - d. vont afficher 10 nombres

7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
8. `printf("%i", 'A')`
- Affiche le caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 54
 - 58
 - 68
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(a, b) ;`
 - `echange(*a, *b) ;`
 - `echange(&a, &b) ;`
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
13. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
14. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair

15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

16. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur FAUX
- c. a pour valeur VRAI
- d. n'a pas de valeur

17. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- c. Diminue de 1 la valeur de a
- d. Réalise une affectation

18. Après `char c; c='a'; c=c+1;`

- a. c vaut `'a'`
- b. Un message d'erreur s'affiche
- c. c vaut `'b'`
- d. c vaut `'A'`

19. `if` est :

- a. Un identificateur du langage C
- b. Un opérateur du langage C
- c. Un mot-clef du langage C
- d. Une commande qu'on tape dans la fenêtre de commande

20. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier pair
- b. N'affiche rien (quelque soit la valeur de a)
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand a est un entier impair

Sujet n° 88

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. entraîne l'affichage d'un message d'erreur
2. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère stocké dans la variable A
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 54
 - c. 68
 - d. 58
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
5. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
6. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand `a` est un entier impair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier pair
7. Après `char c; c='a'; c=c+1;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut 'A'
 - c. `c` vaut 'b'
 - d. `c` vaut 'a'

8. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont afficher 11 nombres
b. vont afficher 10 nombres
c. vont afficher 9 nombres
d. vont boucler indéfiniment
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
a. Permettent de d'affecter à `t[10]` la valeur 0
b. Permettent de d'affecter à `t[10]` la valeur 100
c. Permettent de d'affecter à `t[10]` la valeur 45
d. Permettent de d'affecter à `t[10]` la valeur 10
10. L'expression `a<=1`
a. Utilise les opérateurs `<` et `=`
b. Diminue de 1 la valeur de `a`
c. Réalise une affectation
d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
a. l'adresse de la variable `a`
b. n'a pas de sens
c. la valeur de `a` tout simplement
d. ce vers quoi pointe le pointeur `a`
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
a. `echange(&a, &b) ;`
b. `echange(*a, *b) ;`
c. `echange(a, b) ;`
d. `echange(a++, b++) ;`
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
a. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
b. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
14. `if` est :
a. Un mot-clef du langage C
b. Un identificateur du langage C
c. Un opérateur du langage C
d. Une commande qu'on tape dans la fenêtre de commande

15. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier pair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Affiche bonjour quand a est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
16. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de a quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour quelque soit a
 - d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
17. `printf("%c", 'A')`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
18. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101111
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010100110
20. L'adresse d'une variable c'est :
- a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. le contenu de la variable
 - c. ce vers quoi pointe la variable
 - d. l'adresse d'un pointeur sur la variable

Sujet n° 89

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'b'`
 - b. `c` vaut `'A'`
 - c. `c` vaut `'a'`
 - d. Un message d'erreur s'affiche
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
3. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 9 nombres
 - b. vont afficher 10 nombres
 - c. vont afficher 11 nombres
 - d. vont boucler indéfiniment
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(&a, &b) ;`
 - b. `echange(*a, *b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(a, b) ;`
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 54
 - c. 68
 - d. 58
6. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`

7. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
8. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de *a*
9. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de *a* tout simplement
 - b. ce vers quoi pointe le pointeur *a*
 - c. n'a pas de sens
 - d. l'adresse de la variable *a*
10. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 0
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
12. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur FAUX
 - d. n'a pas de valeur
13. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Affiche bonjour quand *a* est un entier pair
 - d. Affiche bonjour quand *a* est un entier impair
14. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. ce vers quoi pointe la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké

15. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier pair
16. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
17. `if` est :
- Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un opérateur du langage C
18. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010101000
 - 111011010100000
 - 111011010100110
20. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Augmente la valeur de a quelque soit a

Sujet n° 90

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le caractère 'A'
2. `if` est :
 - a. Un mot-clef du langage C
 - b. Un identificateur du langage C
 - c. Un opérateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
 - a. `echange(a, b) ;`
 - b. `echange(a++, b++) ;`
 - c. `echange(&a, &b) ;`
 - d. `echange(*a, *b) ;`
4. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. l'adresse d'un pointeur sur la variable
 - d. le contenu de la variable
5. `if (a%2 == 0) printf("bonjour") ;`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier impair
6. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. a pour valeur VRAI
 - c. n'a pas de valeur
 - d. entraîne l'affichage d'un message d'erreur
7. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

8. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut 'A'
 - `c` vaut 'a'
 - Un message d'erreur s'affiche
 - `c` vaut 'b'
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
10. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
11. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de `a`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010100110
 - 111011010101000
 - 111011010100000
14. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
15. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
- vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont afficher 9 nombres

16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
 - b. 68
 - c. 54
 - d. 58
18. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand `a` est un entier pair
 - b. Affiche bonjour quand `a` est un entier impair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
20. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

Sujet n° 91

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. n'a pas de valeur
 - d. a pour valeur FAUX
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. l'adresse de la variable `a`
 - d. ce vers quoi pointe le pointeur `a`
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
4. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
5. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le caractère 'A'
6. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Diminue de 1 la valeur de `a`
 - c. Réalise une affectation
 - d. Utilise les opérateurs `<` et `=`

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(&a, &b) ;`
- c. `echange(a, b) ;`
- d. `echange(*a, *b) ;`

8. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le code ASCII du caractère stocké dans la variable A

9. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont afficher 9 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment

10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. l'adresse de la variable `a`
- b. la valeur de `a` tout simplement
- c. n'a pas de sens
- d. ce vers quoi pointe le pointeur `a`

11. `if` est :

- a. Un opérateur du langage C
- b. Un mot-clef du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un identificateur du langage C

12. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 62
- c. 54
- d. 58

13. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 10
- b. Permettent de d'affecter à `t[10]` la valeur 100
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 45

14. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier pair
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

15. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier pair
 - Affiche bonjour quand a est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
16. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de a quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
17. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
18. Après `char c; c='a'; c=c+1;`
- c vaut 'b'
 - c vaut 'A'
 - Un message d'erreur s'affiche
 - c vaut 'a'
19. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

Sujet n° 92

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - b. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - d. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
3. L'expression `a<=1`
 - a. Réalise une affectation
 - b. Utilise les opérateurs `<` et `=`
 - c. Diminue de 1 la valeur de a
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. ce vers quoi pointe le pointeur `a`
5. Après `char c ; c='a' ; c=c+1 ;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut `'b'`
 - c. `c` vaut `'A'`
 - d. `c` vaut `'a'`
6. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont boucler indéfiniment
 - b. vont afficher 9 nombres
 - c. vont afficher 11 nombres
 - d. vont afficher 10 nombres

7. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - n'a pas de valeur
 - a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
8. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
10. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010101111
 - 111011010100110
 - 111011010100000
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(a, b) ;`
 - `echange(&a, &b) ;`
13. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
14. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'

15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
 - b. 54
 - c. 58
 - d. 68
16. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 100
17. `if (a%2 == 0) printf("bonjour") ;`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand `a` est un entier impair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Affiche bonjour quand `a` est un entier pair
18. `printf("%i", A)`
- a. Affiche le caractère '`A`'
 - b. Affiche le code ASCII du caractère '`A`'
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
20. `if` est :
- a. Un mot-clef du langage C
 - b. Un identificateur du langage C
 - c. Un opérateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande

Sujet n° 93

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier pair
2. `if` est :
 - a. Un opérateur du langage C
 - b. Un mot-clef du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un identificateur du langage C
3. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère ' A '
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère ' A '
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010100110
 - c. 111011010101111
 - d. 111011010101000
5. Après `char c; c='a'; c=c+1;`
 - a. c vaut ' b '
 - b. Un message d'erreur s'affiche
 - c. c vaut ' A '
 - d. c vaut ' a '
6. `if (a%2 = 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier impair
 - c. Affiche bonjour quand a est un entier pair
 - d. N'affiche rien (quelque soit la valeur de a)
7. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

8. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
 - n'a pas de valeur
 - a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
9. L'expression `a<=1`
- Réalise une affectation
 - Diminue de 1 la valeur de a
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
10. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
11. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
 - `echange(a, b) ;`
 - `echange(&a, &b) ;`
14. `printf("%i", 'A')`
- Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
15. `printf("%c", 'A')`
- Affiche le caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`

16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 58
 - b. 68
 - c. 54
 - d. 62
17. Les instructions `i=0 ;`
- ```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- a. vont afficher 11 nombres
  - b. vont afficher 9 nombres
  - c. vont afficher 10 nombres
  - d. vont boucler indéfiniment
18. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - b. Augmente la valeur de  $a$  quelque soit  $a$
  - c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
20. `printf("%i", A)`
- a. Affiche le caractère '`A`'
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère '`A`'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

# Sujet n° 94

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le contenu de la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
3. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
4. `if (a%2 == 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier pair
5. `if (a%2 = 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier impair
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(&a, &b);`
  - b. `echange(a++, b++);`
  - c. `echange(*a, *b);`
  - d. `echange(a, b);`
7. `printf("%c", A)`
  - a. Affiche le caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
  - 62
  - 58
  - 54
9. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
10. `if` est :
- Un opérateur du langage C
  - Un identificateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
11. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
12. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - n'a pas de sens
  - l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
14. Les instructions
- ```

i=0;
while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 11 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
15. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010101111
 - 111011010100000
 - 111011010100110

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

17. Après `char c; c='a'; c=c+1;`

- a. `c` vaut `'A'`
- b. Un message d'erreur s'affiche
- c. `c` vaut `'b'`
- d. `c` vaut `'a'`

18. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur VRAI
- c. a pour valeur FAUX
- d. n'a pas de valeur

19. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère `'A'`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère `'A'`

20. L'expression `a<=1`

- a. Diminue de 1 la valeur de `a`
- b. Réalise une affectation
- c. Utilise les opérateurs `<` et `=`
- d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon

Sujet n° 95

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. le contenu de la variable
2. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
3. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'A'`
 - b. `c` vaut `'a'`
 - c. Un message d'erreur s'affiche
 - d. `c` vaut `'b'`
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010101111
 - c. 111011010101000
 - d. 111011010100000
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
6. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 10 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 11 nombres
 - d. vont afficher 9 nombres

7. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - a pour valeur FAUX
8. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 68
 - 54
 - 58
10. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère '`A`'
 - Affiche le code ASCII du caractère '`A`'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
12. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un mot-clef du langage C
13. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
 - Affiche bonjour quand `a` est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
14. L'expression `a<=1`
- Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de `a`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon

15. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier pair
16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. la valeur de `a` tout simplement
17. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`
18. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Augmente la valeur de a quelque soit a
 - b. Affiche bonjour quelque soit a
 - c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - d. Affiche bonjour et augmente la valeur de a quelque soit a
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b);`
 - b. `echange(a++, b++);`
 - c. `echange(&a, &b);`
 - d. `echange(*a, *b);`
20. `printf("%c", 'A')`
- a. Affiche le caractère `'A'`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`

Sujet n° 96

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 9 nombres
- d. vont afficher 10 nombres

2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101111
- c. 111011010101000
- d. 111011010100000

3. `printf("%i", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le code ASCII du caractère stocké dans la variable A

4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. l'adresse de la variable `a`
- d. ce vers quoi pointe le pointeur `a`

5. L'expression `a<=1`

- a. Réalise une affectation
- b. Utilise les opérateurs `<` et `=`
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Diminue de 1 la valeur de `a`

6. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de `a`)
- b. Affiche bonjour quand `a` est un entier pair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand `a` est un entier impair

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b) ;`
- b. `echange(a, b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(&a, &b) ;`

8. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 62
- b. 54
- c. 68
- d. 58

9. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

10. `if` est :

- a. Un opérateur du langage C
- b. Un mot-clef du langage C
- c. Un identificateur du langage C
- d. Une commande qu'on tape dans la fenêtre de commande

11. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut 'A'
- b. `c` vaut 'a'
- c. Un message d'erreur s'affiche
- d. `c` vaut 'b'

12. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier impair
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier pair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

13. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- b. Augmente la valeur de `a` quelque soit `a`
- c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- d. Affiche bonjour quelque soit `a`

14. `printf("%i", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
17. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - le contenu de la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
18. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur FAUX
20. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10

Sujet n° 97

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
2. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère 'A'
3. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Augmente la valeur de a quelque soit a
 - b. Affiche bonjour quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
4. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur FAUX
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. ce vers quoi pointe le pointeur `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens
6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
7. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Diminue de 1 la valeur de a
 - c. Réalise une affectation
 - d. Utilise les opérateurs `<` et `=`

8. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
9. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier pair
 - Affiche bonjour quand *a* est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
11. `if` est :
- Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un opérateur du langage C
12. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
13. Après `char c; c='a'; c=c+1;`
- c* vaut 'A'
 - c* vaut 'b'
 - Un message d'erreur s'affiche
 - c* vaut 'a'
14. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a, b);`
 - `echange(&a, &b);`
 - `echange(a++, b++);`
 - `echange(*a, *b);`

16. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
 - Affiche bonjour quand a est un entier pair
17. Les instructions `i=0;`
- ```

while(i<10)
 printf("%i ", i);
 i++;

```
- vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment
18. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 68
  - 62
  - 58
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101111
  - 111011010100110
  - 111011010101000
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

# Sujet n° 98

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'A'`
  - b. `c` vaut `'a'`
  - c. Un message d'erreur s'affiche
  - d. `c` vaut `'b'`
2. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 62
  - c. 68
  - d. 58
4. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(*a, *b) ;`
  - b. `echange(a++, b++) ;`
  - c. `echange(&a, &b) ;`
  - d. `echange(a, b) ;`
6. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont boucler indéfiniment
  - b. vont afficher 9 nombres
  - c. vont afficher 10 nombres
  - d. vont afficher 11 nombres



7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le caractère 'A'
8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
9. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier pair
  - Affiche bonjour quand a est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de a)
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
  - 111011010101111
  - 111011010100000
  - 111011010100110
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 0
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur VRAI
  - a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur

15. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - le contenu de la variable
16. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
17. L'expression `a<=1`
- A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Diminue de 1 la valeur de  $a$
  - Utilise les opérateurs `<` et `=`
  - Réalise une affectation
18. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier impair
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
20. `if` est :
- Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
  - Un identificateur du langage C

# Sujet n° 99

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur VRAI
  - b. a pour valeur FAUX
  - c. entraîne l'affichage d'un message d'erreur
  - d. n'a pas de valeur
2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
3. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
4. Après `char c; c='a'; c=c+1;`
  - a. `c` vaut `'b'`
  - b. `c` vaut `'A'`
  - c. Un message d'erreur s'affiche
  - d. `c` vaut `'a'`
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 58
  - c. 54
  - d. 62
6. `printf("%c", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`
7. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - c. Affiche bonjour quelque soit `a`
  - d. Augmente la valeur de `a` quelque soit `a`

8. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier impair
  - Affiche bonjour quand  $a$  est un entier pair
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b);`
  - `echange(a++, b++);`
  - `echange(*a, *b);`
  - `echange(&a, &b);`
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010101000
  - 111011010100000
  - 111011010100110
12. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
13. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
  - le contenu de la variable
14. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
15. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`

16. L'expression `a<=1`
- a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Diminue de 1 la valeur de  $a$
  - c. Réalise une affectation
  - d. Utilise les opérateurs `<` et `=`
17. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le caractère `'A'`
18. `if` est :
- a. Un identificateur du langage C
  - b. Un opérateur du langage C
  - c. Un mot-clef du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
19. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont boucler indéfiniment
 - b. vont afficher 9 nombres
 - c. vont afficher 11 nombres
 - d. vont afficher 10 nombres
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. la valeur de `a` tout simplement

Sujet n° 100

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010100110
 - c. 111011010101000
 - d. 111011010101111
3. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. l'adresse d'un pointeur sur la variable
 - d. le contenu de la variable
4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
5. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
7. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 100
 - d. Permettent de d'affecter à `t[10]` la valeur 0

8. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier pair
 - N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - a pour valeur FAUX
 - a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
10. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
 - N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
11. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
 - 62
 - 54
 - 58
13. Les instructions
- ```
i=0;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
- `echange(a++, b++)` ;
  - `echange(a, b)` ;
  - `echange(&a, &b)` ;
  - `echange(*a, *b)` ;
15. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$

16. L'expression `a<=1`
- a. Réalise une affectation
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Diminue de 1 la valeur de  $a$
17. Après `char c ; c='a' ; c=c+1 ;`
- a.  $c$  vaut `'a'`
  - b.  $c$  vaut `'b'`
  - c. Un message d'erreur s'affiche
  - d.  $c$  vaut `'A'`
18. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$
19. `if` est :
- a. Un mot-clef du langage C
  - b. Un identificateur du langage C
  - c. Un opérateur du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. n'a pas de sens

# Sujet n° 101

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. l'adresse de la variable `a`
  - d. ce vers quoi pointe le pointeur `a`
2. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand `a` est un entier pair
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
3. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
  - a. vont afficher 9 nombres
  - b. vont afficher 10 nombres
  - c. vont boucler indéfiniment
  - d. vont afficher 11 nombres
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100110
  - b. 111011010101111
  - c. 111011010101000
  - d. 111011010100000
5. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
6. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(*a, *b) ;`
- c. `echange(&a, &b) ;`
- d. `echange(a, b) ;`

8. `printf("%i", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère 'A'

9. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur VRAI
- c. a pour valeur FAUX
- d. entraîne l'affichage d'un message d'erreur

10. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 62
- b. 58
- c. 68
- d. 54

11. `if (a%2 == 0) printf("bonjour") ;`

- a. N'affiche rien (quelque soit la valeur de `a`)
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand `a` est un entier impair
- d. Affiche bonjour quand `a` est un entier pair

12. `printf("%i", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

13. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 100
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 0

14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
- b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
- c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
- d. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`



15. L'expression `a<=1`
- a. Diminue de 1 la valeur de  $a$
  - b. Réalise une affectation
  - c. Utilise les opérateurs `<` et `=`
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. l'adresse de la variable `a`
  - d. la valeur de `a` tout simplement
17. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
18. `if` est :
- a. Un identificateur du langage C
  - b. Un mot-clef du langage C
  - c. Un opérateur du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
19. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère `'A'`
20. Après `char c; c='a'; c=c+1;`
- a.  $c$  vaut `'A'`
  - b. Un message d'erreur s'affiche
  - c.  $c$  vaut `'a'`
  - d.  $c$  vaut `'b'`

# Sujet n° 102

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 62
  - c. 58
  - d. 54
2. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le code ASCII du caractère stocké dans la variable A
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101000
  - b. 111011010100110
  - c. 111011010100000
  - d. 111011010101111
4. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur VRAI
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
6. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 45
7. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable A
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le caractère 'A'

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - n'a pas de sens
9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
10. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - `c` vaut `'A'`
  - `c` vaut `'a'`
  - `c` vaut `'b'`
11. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
  - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
13. L'adresse d'une variable c'est :
- le contenu de la variable
  - l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
14. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
  - Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de `a`)
15. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont afficher 11 nombres
  - vont boucler indéfiniment
  - vont afficher 9 nombres
  - vont afficher 10 nombres

16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier pair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
17. L'expression `a<=1`
- a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Réalise une affectation
  - c. Utilise les opérateurs `<` et `=`
  - d. Diminue de 1 la valeur de  $a$
18. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère 'A'
19. `if` est :
- a. Un mot-clef du langage C
  - b. Une commande qu'on tape dans la fenêtre de commande
  - c. Un opérateur du langage C
  - d. Un identificateur du langage C
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
- a. `echange(a++, b++) ;`
  - b. `echange(*a, *b) ;`
  - c. `echange(&a, &b) ;`
  - d. `echange(a, b) ;`

# Sujet n° 103

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
  - a. Utilise les opérateurs `<` et `=`
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Réalise une affectation
  - d. Diminue de 1 la valeur de  $a$
2. `printf("%i", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 54
  - c. 68
  - d. 62
4. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un identificateur du langage C
  - c. Un opérateur du langage C
  - d. Un mot-clef du langage C
5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100110
  - b. 111011010101111
  - c. 111011010101000
  - d. 111011010100000
6. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Affiche bonjour quelque soit  $a$
  - b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - c. Augmente la valeur de  $a$  quelque soit  $a$
  - d. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
7. `printf("%c", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère 'A'

8. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'a'`
  - b. Un message d'erreur s'affiche
  - c. `c` vaut `'b'`
  - d. `c` vaut `'A'`
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de `a` tout simplement
  - c. l'adresse de la variable `a`
  - d. ce vers quoi pointe le pointeur `a`
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. n'a pas de sens
11. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. le contenu de la variable
12. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier pair
13. `if (a%2 = 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier impair
  - b. Affiche bonjour quand `a` est un entier pair
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
14. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 100
15. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
16. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`

17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(*a, *b);`
- c. `echange(a, b);`
- d. `echange(a++, b++);`

18. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. n'a pas de valeur
- c. a pour valeur VRAI
- d. a pour valeur FAUX

19. Les instructions `i=0;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 10 nombres
- b. vont afficher 9 nombres
- c. vont boucler indéfiniment
- d. vont afficher 11 nombres

20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

# Sujet n° 104

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'A'`
  - b. Un message d'erreur s'affiche
  - c. `c` vaut `'a'`
  - d. `c` vaut `'b'`
2. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. ce vers quoi pointe le pointeur `a`
  - c. n'a pas de sens
  - d. la valeur de `a` tout simplement
4. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 68
  - c. 62
  - d. 58
5. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le contenu de la variable
6. `if` est :
  - a. Un identificateur du langage C
  - b. Un opérateur du langage C
  - c. Un mot-clef du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
7. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand `a` est un entier impair
  - d. N'affiche rien (quelque soit la valeur de `a`)

8. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont afficher 11 nombres  
b. vont afficher 9 nombres  
c. vont boucler indéfiniment  
d. vont afficher 10 nombres
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :  
a. ce vers quoi pointe le pointeur `a`  
b. l'adresse de la variable `a`  
c. n'a pas de sens  
d. la valeur de `a` tout simplement
10. `printf("%i", 'A')`  
a. Affiche le code ASCII du caractère `'A'`  
b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`  
c. Affiche le code ASCII du caractère stocké dans la variable `A`  
d. Affiche le caractère `'A'`
11. `printf("%c", 'A')`  
a. Affiche le code ASCII du caractère `'A'`  
b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`  
c. Affiche le code ASCII du caractère stocké dans la variable `A`  
d. Affiche le caractère `'A'`
12. `if (a%2 == 0) printf("bonjour") ;`  
a. Affiche bonjour quand `a` est un entier impair  
b. N'affiche rien (quelque soit la valeur de `a`)  
c. Déclenche le message d'erreur `invalid lvalue in assignment`  
d. Affiche bonjour quand `a` est un entier pair
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :  
a. `echange(a, b) ;`  
b. `echange(a++, b++) ;`  
c. `echange(&a, &b) ;`  
d. `echange(*a, *b) ;`
14. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`  
a. Permettent de d'affecter à `t[10]` la valeur 100  
b. Permettent de d'affecter à `t[10]` la valeur 0  
c. Permettent de d'affecter à `t[10]` la valeur 10  
d. Permettent de d'affecter à `t[10]` la valeur 45
15. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :  
a. 111011010101000  
b. 111011010101111  
c. 111011010100110  
d. 111011010100000

16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
  - b. entraîne l'affichage d'un message d'erreur
  - c. n'a pas de valeur
  - d. a pour valeur VRAI
17. L'expression `a<=1`
- a. Réalise une affectation
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Diminue de 1 la valeur de  $a$
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
20. `if (a<5) printf("Bonjour"); a=a+1;`
- a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour quelque soit  $a$
  - d. Augmente la valeur de  $a$  quelque soit  $a$



# Sujet n° 105

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le caractère 'A'
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. n'a pas de valeur
  - c. a pour valeur FAUX
  - d. a pour valeur VRAI
3. `printf("%c", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable A
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
4. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 68
  - c. 62
  - d. 58
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
6. On suppose que a a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur a
  - c. l'adresse de la variable a
  - d. la valeur de a tout simplement
7. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand a est un entier pair
  - b. Affiche bonjour quand a est un entier impair
  - c. N'affiche rien (quelque soit la valeur de a)
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`

8. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont boucler indéfiniment  
b. vont afficher 10 nombres  
c. vont afficher 9 nombres  
d. vont afficher 11 nombres
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :  
a. 111011010100110  
b. 111011010100000  
c. 111011010101111  
d. 111011010101000
10. `if (a<5) printf("Bonjour") ; a=a+1 ;`  
a. Augmente la valeur de  $a$  quelque soit  $a$   
b. Affiche bonjour quelque soit  $a$   
c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$   
d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
11. `printf("%i", 'A')`  
a. Affiche le code ASCII du caractère stocké dans la variable  $A$   
b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$   
c. Affiche le caractère 'A'  
d. Affiche le code ASCII du caractère 'A'
12. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`  
a. Permettent de d'affecter à  $t[10]$  la valeur 10  
b. Permettent de d'affecter à  $t[10]$  la valeur 45  
c. Permettent de d'affecter à  $t[10]$  la valeur 100  
d. Permettent de d'affecter à  $t[10]$  la valeur 0
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :  
a. `echange(a, b) ;`  
b. `echange(&a, &b) ;`  
c. `echange(a++, b++) ;`  
d. `echange(*a, *b) ;`
14. `if (a%2 == 0) printf("bonjour") ;`  
a. Affiche bonjour quand  $a$  est un entier pair  
b. N'affiche rien (quelque soit la valeur de  $a$ )  
c. Déclenche le message d'erreur `invalid lvalue in assignment`  
d. Affiche bonjour quand  $a$  est un entier impair
15. L'expression `a<=1`  
a. Réalise une affectation  
b. Utilise les opérateurs `<` et `=`  
c. Diminue de 1 la valeur de  $a$   
d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon

16. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'A'`
  - b. `c` vaut `'a'`
  - c. Un message d'erreur s'affiche
  - d. `c` vaut `'b'`
17. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
  - b. le contenu de la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable
18. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
  - b. la valeur de `a` tout simplement
  - c. l'adresse de la variable `a`
  - d. ce vers quoi pointe le pointeur `a`
20. `if` est :
- a. Un identificateur du langage C
  - b. Une commande qu'on tape dans la fenêtre de commande
  - c. Un mot-clef du langage C
  - d. Un opérateur du langage C

# Sujet n° 106

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. n'a pas de valeur
  - b. a pour valeur VRAI
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur FAUX
2. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 100
3. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'a'`
  - b. `c` vaut `'A'`
  - c. `c` vaut `'b'`
  - d. Un message d'erreur s'affiche
4. `if` est :
  - a. Un mot-clef du langage C
  - b. Une commande qu'on tape dans la fenêtre de commande
  - c. Un opérateur du langage C
  - d. Un identificateur du langage C
5. `printf("%i", 'A')`
  - a. Affiche le caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère `'A'`
6. `printf("%c", A)`
  - a. Affiche le caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère `'A'`



7. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`  
a. vont afficher 9 nombres  
b. vont afficher 10 nombres  
c. vont boucler indéfiniment  
d. vont afficher 11 nombres
8. `if (a<5) printf("Bonjour") ; a=a+1 ;`  
a. Augmente la valeur de  $a$  quelque soit  $a$   
b. Affiche bonjour quelque soit  $a$   
c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$   
d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
9. `if (a%2 == 0) printf("bonjour") ;`  
a. Affiche bonjour quand  $a$  est un entier impair  
b. Déclenche le message d'erreur `invalid lvalue in assignment`  
c. Affiche bonjour quand  $a$  est un entier pair  
d. N'affiche rien (quelque soit la valeur de  $a$ )
10. `printf("%c", 'A')`  
a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$   
b. Affiche le code ASCII du caractère 'A'  
c. Affiche le code ASCII du caractère stocké dans la variable  $A$   
d. Affiche le caractère 'A'
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :  
a. `echange(*a, *b) ;`  
b. `echange(a, b) ;`  
c. `echange(a++, b++) ;`  
d. `echange(&a, &b) ;`
12. On suppose que  $a$  a été déclarée par `int a`. L'expression `&a` a pour valeur :  
a. la valeur de  $a$  tout simplement  
b. ce vers quoi pointe le pointeur  $a$   
c. l'adresse de la variable  $a$   
d. n'a pas de sens
13. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :  
a. `111011010101111`  
b. `111011010101000`  
c. `111011010100000`  
d. `111011010100110`
14. `if (a%2 = 0) printf("bonjour") ;`  
a. N'affiche rien (quelque soit la valeur de  $a$ )  
b. Déclenche le message d'erreur `invalid lvalue in assignment`  
c. Affiche bonjour quand  $a$  est un entier impair  
d. Affiche bonjour quand  $a$  est un entier pair

15. L'adresse d'une variable c'est :
- le contenu de la variable
  - l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 58
  - 68
17. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère '`A`'
  - Affiche le code ASCII du caractère '`A`'
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
20. L'expression `a<=1`
- Réalise une affectation
  - Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de `a`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon

# Sujet n° 107

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b) ;`
- b. `echange(a, b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(&a, &b) ;`

2. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 100

3. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. N'affiche rien (quelque soit la valeur de `a`)
- d. Affiche bonjour quand `a` est un entier pair

4. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

5. L'expression `a<=1`

- a. Réalise une affectation
- b. Utilise les opérateurs `<` et `=`
- c. Diminue de 1 la valeur de `a`
- d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon

6. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

7. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour quelque soit `a`
- b. Augmente la valeur de `a` quelque soit `a`
- c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`

8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 68
  - 54
  - 62
9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens
  - l'adresse de la variable `a`
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
  - 111011010100000
  - 111011010101111
  - 111011010100110
11. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
12. `printf("%c", A)`
- Affiche le caractère '`A`'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
13. Après `char c; c='a'; c=c+1;`
- `c` vaut '`a`'
  - `c` vaut '`A`'
  - Un message d'erreur s'affiche
  - `c` vaut '`b`'
14. Les instructions
- ```

i=0;
while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
15. `if` est :
- Un identificateur du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C

16. L'adresse d'une variable c'est :
- a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. ce vers quoi pointe la variable
 - c. le contenu de la variable
 - d. l'adresse d'un pointeur sur la variable
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. la valeur de `a` tout simplement
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
 - b. n'a pas de valeur
 - c. a pour valeur VRAI
 - d. entraîne l'affichage d'un message d'erreur
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
20. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le caractère `'A'`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

Sujet n° 108

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. la valeur de `a` tout simplement
- c. l'adresse de la variable `a`
- d. n'a pas de sens

2. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 10 nombres
- b. vont boucler indéfiniment
- c. vont afficher 9 nombres
- d. vont afficher 11 nombres

3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 10
- b. Permettent de d'affecter à `t[10]` la valeur 100
- c. Permettent de d'affecter à `t[10]` la valeur 45
- d. Permettent de d'affecter à `t[10]` la valeur 0

4. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Réalise une affectation
- c. Utilise les opérateurs `<` et `=`
- d. Diminue de 1 la valeur de `a`

5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. l'adresse de la variable `a`
- c. ce vers quoi pointe le pointeur `a`
- d. la valeur de `a` tout simplement

6. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. N'affiche rien (quelque soit la valeur de `a`)
- d. Affiche bonjour quand `a` est un entier pair

7. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur FAUX
 - a pour valeur VRAI
8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
 - `echange(&a, &b) ;`
 - `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
9. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
10. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
11. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :
- `111011010100000`
 - `111011010101111`
 - `111011010101000`
 - `111011010100110`
12. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier pair
13. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
14. `if` est :
- Un mot-clef du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C

15. L'adresse d'une variable c'est :
- le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
16. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
17. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
18. Après `char c; c='a'; c=c+1;`
- c vaut 'b'
 - Un message d'erreur s'affiche
 - c vaut 'A'
 - c vaut 'a'
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 68
 - 62
 - 54

Sujet n° 109

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 62
 - c. 58
 - d. 54
2. `printf("%c", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010101000
 - c. 111011010100110
 - d. 111011010101111
4. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand *a* est un entier pair
 - d. Affiche bonjour quand *a* est un entier impair
5. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur *a*
 - b. n'a pas de sens
 - c. la valeur de *a* tout simplement
 - d. l'adresse de la variable *a*
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
 - a. `echange(*a, *b);`
 - b. `echange(a, b);`
 - c. `echange(a++, b++);`
 - d. `echange(&a, &b);`

7. L'expression `a<=1`
 - a. Diminue de 1 la valeur de *a*
 - b. Utilise les opérateurs `<` et `=`
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Réalise une affectation
8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 45
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - b. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - c. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - d. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
10. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
11. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur *a*
 - b. n'a pas de sens
 - c. la valeur de *a* tout simplement
 - d. l'adresse de la variable *a*
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - b. Augmente la valeur de *a* quelque soit *a*
 - c. Affiche bonjour quelque soit *a*
 - d. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
13. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand *a* est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier pair
14. `if` est :
 - a. Un identificateur du langage C
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande

15. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'b'`
 - b. `c` vaut `'a'`
 - c. Un message d'erreur s'affiche
 - d. `c` vaut `'A'`
16. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
 - b. le contenu de la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
17. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`
18. `printf("%i", A)`
- a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur FAUX
 - d. n'a pas de valeur
20. Les instructions `i=0 ;`
- ```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- a. vont afficher 11 nombres
  - b. vont afficher 9 nombres
  - c. vont afficher 10 nombres
  - d. vont boucler indéfiniment

# Sujet n° 110

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 11 nombres
- c. vont afficher 10 nombres
- d. vont afficher 9 nombres

2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010100110
- c. 111011010101111
- d. 111011010101000

3. `printf("%i", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

4. L'expression `a<=1`

- a. Réalise une affectation
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Utilise les opérateurs < et =
- d. Diminue de 1 la valeur de a

5. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le code ASCII du caractère stocké dans la variable A

6. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de a)
- b. Affiche bonjour quand a est un entier pair
- c. Affiche bonjour quand a est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

7. `if` est :
- Un mot-clef du langage C
  - Un identificateur du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
  - `echange(*a, *b) ;`
  - `echange(&a, &b) ;`
  - `echange(a, b) ;`
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
  - a pour valeur VRAI
  - n'a pas de valeur
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
11. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
  - Affiche bonjour quand `a` est un entier impair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
13. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
14. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
  - `c` vaut `'A'`
  - `c` vaut `'b'`
  - `c` vaut `'a'`

15. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. la valeur de `a` tout simplement
16. `printf("%c", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
17. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. l'adresse d'un pointeur sur la variable
  - d. le contenu de la variable
18. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour quelque soit `a`
  - b. Augmente la valeur de `a` quelque soit `a`
  - c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
19. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 54
  - b. 62
  - c. 58
  - d. 68

# Sujet n° 111

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101111
  - c. 111011010101000
  - d. 111011010100110
3. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Augmente la valeur de  $a$  quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
4. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$
5. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
  - a. vont afficher 9 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 10 nombres
  - d. vont afficher 11 nombres
6. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le contenu de la variable

7. `if` est :
- Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
  - Un mot-clef du langage C
8. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
9. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
10. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Diminue de 1 la valeur de `a`
11. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
12. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
13. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
  - a pour valeur VRAI
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b);`
  - `echange(a, b);`
  - `echange(a++, b++);`
  - `echange(*a, *b);`

15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 58
  - 68
  - 54
16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
17. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
20. Après `char c; c='a'; c=c+1;`
- `c` vaut `'a'`
  - `c` vaut `'b'`
  - Un message d'erreur s'affiche
  - `c` vaut `'A'`

# Sujet n° 112

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier pair
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
2. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
  - a. vont afficher 9 nombres
  - b. vont afficher 11 nombres
  - c. vont afficher 10 nombres
  - d. vont boucler indéfiniment
3. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
4. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 10
5. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. l'adresse de la variable `a`
  - c. n'a pas de sens
  - d. la valeur de `a` tout simplement
6. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'b'`
  - b. Un message d'erreur s'affiche
  - c. `c` vaut `'A'`
  - d. `c` vaut `'a'`

7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
8. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit *a*
  - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - Augmente la valeur de *a* quelque soit *a*
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010101111
  - 111011010100000
  - 111011010101000
10. `printf("%i", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 54
  - 68
  - 62
12. `if` est :
- Un identificateur du langage C
  - Un mot-clef du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
13. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
  - Affiche bonjour quand *a* est un entier pair
  - N'affiche rien (quelque soit la valeur de *a*)
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

15. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
16. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
  - a pour valeur VRAI
  - n'a pas de valeur
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(&a, &b) ;`
  - `echange(a, b) ;`
  - `echange(*a, *b) ;`
  - `echange(a++, b++) ;`
18. `printf("%i", A)`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
19. L'expression `a<=1`
- A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
  - Diminue de 1 la valeur de *a*
  - Utilise les opérateurs `<` et `=`
20. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de *a* tout simplement
  - l'adresse de la variable *a*
  - n'a pas de sens
  - ce vers quoi pointe le pointeur *a*



# Sujet n° 113

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
2. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de *a* tout simplement
  - c. l'adresse de la variable *a*
  - d. ce vers quoi pointe le pointeur *a*
3. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
4. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
5. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Utilise les opérateurs `<` et `=`
  - c. Diminue de 1 la valeur de *a*
  - d. Réalise une affectation
6. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. ce vers quoi pointe le pointeur *a*
  - b. n'a pas de sens
  - c. l'adresse de la variable *a*
  - d. la valeur de *a* tout simplement
7. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101000
  - c. 111011010100110
  - d. 111011010101111

8. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - `c` vaut `'a'`
  - `c` vaut `'A'`
  - `c` vaut `'b'`
9. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit `a`
  - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
10. `if` est :
- Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
  - Un mot-clef du langage C
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 58
  - 54
  - 68
12. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
13. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```
- vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
 - vont afficher 11 nombres
14. L'adresse d'une variable c'est :
- le contenu de la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
15. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`

16. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 100
17. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand *a* est un entier pair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand *a* est un entier impair
 - d. N'affiche rien (quelque soit la valeur de *a*)
18. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- a. `echange(a, b);`
 - b. `echange(a++, b++);`
 - c. `echange(&a, &b);`
 - d. `echange(*a, *b);`
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
20. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Affiche bonjour quand *a* est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand *a* est un entier pair

Sujet n° 114

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Affiche bonjour quand a est un entier pair
 - d. Affiche bonjour quand a est un entier impair
2. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. a pour valeur VRAI
 - c. entraîne l'affichage d'un message d'erreur
 - d. n'a pas de valeur
3. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
4. Les instructions

```
i=0 ;
while(i<10)
    printf("%i ", i);
    i++;
```

 - a. vont boucler indéfiniment
 - b. vont afficher 9 nombres
 - c. vont afficher 10 nombres
 - d. vont afficher 11 nombres
5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100000
 - b. 111011010101111
 - c. 111011010101000
 - d. 111011010100110
6. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. le contenu de la variable
 - d. l'adresse d'un pointeur sur la variable

7. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
 - Un message d'erreur s'affiche
 - `c` vaut `'A'`
 - `c` vaut `'b'`
8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
10. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
11. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
 - Un identificateur du langage C
 - Un mot-clef du langage C
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(*a, *b) ;`
 - `echange(a, b) ;`
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
14. `printf("%c", A)`
- Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`

15. L'expression `a<=1`
- a. Réalise une affectation
 - b. Diminue de 1 la valeur de a
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Utilise les opérateurs `<` et `=`
16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 54
 - b. 62
 - c. 58
 - d. 68
17. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
19. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Augmente la valeur de a quelque soit a
20. `printf("%i", A)`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'

Sujet n° 115

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier impair
 - d. Affiche bonjour quand a est un entier pair
2. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
3. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
4. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
5. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
6. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
 - a. vont boucler indéfiniment
 - b. vont afficher 10 nombres
 - c. vont afficher 11 nombres
 - d. vont afficher 9 nombres

7. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

8. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010100000
- c. 111011010101111
- d. 111011010101000

9. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand *a* est un entier impair
- b. N'affiche rien (quelque soit la valeur de *a*)
- c. Affiche bonjour quand *a* est un entier pair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

10. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. l'adresse de la variable *a*
- c. ce vers quoi pointe le pointeur *a*
- d. la valeur de *a* tout simplement

11. `if (a<5) printf("Bonjour"); a=a+1;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
- b. Affiche bonjour quelque soit *a*
- c. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
- d. Augmente la valeur de *a* quelque soit *a*

12. Après `char c; c='a'; c=c+1;`

- a. Un message d'erreur s'affiche
- b. *c* vaut 'b'
- c. *c* vaut 'a'
- d. *c* vaut 'A'

13. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. entraîne l'affichage d'un message d'erreur
- c. a pour valeur VRAI
- d. a pour valeur FAUX

14. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. l'adresse de la variable *a*
- c. la valeur de *a* tout simplement
- d. ce vers quoi pointe le pointeur *a*

15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a++, b++) ;`
- c. `echange(a, b) ;`
- d. `echange(*a, *b) ;`

16. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- c. Diminue de 1 la valeur de a
- d. Réalise une affectation

17. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 45
- d. Permettent de d'affecter à `t[10]` la valeur 100

18. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. le contenu de la variable
- c. ce vers quoi pointe la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

19. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 62
- b. 68
- c. 58
- d. 54

20. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un opérateur du langage C
- c. Un identificateur du langage C
- d. Un mot-clef du langage C

Sujet n° 116

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 62
 - c. 68
 - d. 54
2. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier pair
 - c. Affiche bonjour quand a est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
3. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le caractère 'A'
4. Après `char c; c='a'; c=c+1;`
 - a. c vaut 'b'
 - b. c vaut 'a'
 - c. c vaut 'A'
 - d. Un message d'erreur s'affiche
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables a et b on doit écrire :
 - a. `echange(*a, *b);`
 - b. `echange(a++, b++);`
 - c. `echange(&a, &b);`
 - d. `echange(a, b);`

7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
10. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
 - Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)
11. Les instructions
- ```
i=0;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
  - vont afficher 11 nombres
12. L'expression `a<=1`
- A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Diminue de 1 la valeur de `a`
  - Utilise les opérateurs `<` et `=`
  - Réalise une affectation
13. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - le contenu de la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
14. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour et augmente la valeur de `a` quelque soit `a`

15. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101000
  - 111011010100110
  - 111011010101111
17. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le caractère 'A'
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
19. `if` est :
- Un identificateur du langage C
  - Un mot-clef du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
20. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
  - a pour valeur FAUX
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur

# Sujet n° 117

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable A
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - d. Affiche le caractère 'A'
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur VRAI
  - c. n'a pas de valeur
  - d. a pour valeur FAUX
3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 100
4. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le contenu de la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
5. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
6. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - c. Affiche le code ASCII du caractère stocké dans la variable A
  - d. Affiche le caractère 'A'
7. Les instructions 

```
i=0 ;
while(i<10)
 printf("%i ", i) ;
i++ ;
```

  - a. vont afficher 10 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 11 nombres
  - d. vont afficher 9 nombres

8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

9. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 62
- b. 54
- c. 58
- d. 68

10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b);`
- b. `echange(a++, b++);`
- c. `echange(&a, &b);`
- d. `echange(*a, *b);`

12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100110
- c. 111011010101111
- d. 111011010100000

13. `if` est :

- a. Un mot-clef du langage C
- b. Un identificateur du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un opérateur du langage C

14. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier impair
- b. Affiche bonjour quand `a` est un entier pair
- c. N'affiche rien (quelque soit la valeur de `a`)
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

15. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand `a` est un entier pair
- d. N'affiche rien (quelque soit la valeur de `a`)



16. L'expression `a<=1`
- Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de  $a$
17. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère `'A'`
18. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
19. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
20. Après `char c; c='a'; c=c+1;`
- $c$  vaut `'A'`
  - Un message d'erreur s'affiche
  - $c$  vaut `'a'`
  - $c$  vaut `'b'`

# Sujet n° 118

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010100000
- c. 111011010100110
- d. 111011010101111

2. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 58
- c. 62
- d. 54

3. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. n'a pas de sens
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

4. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut `'A'`
- b. Un message d'erreur s'affiche
- c. `c` vaut `'b'`
- d. `c` vaut `'a'`

5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b) ;`
- b. `echange(*a, *b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(&a, &b) ;`

6. L'expression `a<=1`

- a. Réalise une affectation
- b. Diminue de 1 la valeur de `a`
- c. Utilise les opérateurs `<` et `=`
- d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon

7. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier pair
8. `if (a%2 = 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
9. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
11. `printf("%c", A)`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
12. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
13. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
  - a pour valeur FAUX
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
14. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$

15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 0
16. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont boucler indéfiniment
  - vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont afficher 11 nombres
17. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
19. `if` est :
- Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
  - Un mot-clef du langage C
20. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
  - l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - le contenu de la variable

# Sujet n° 119

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

2. Après `char c; c='a'; c=c+1;`

- a. `c` vaut `'A'`
- b. Un message d'erreur s'affiche
- c. `c` vaut `'a'`
- d. `c` vaut `'b'`

3. L'expression `a<=1`

- a. Diminue de 1 la valeur de `a`
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Utilise les opérateurs `<` et `=`
- d. Réalise une affectation

4. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

5. `printf("%c", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(&a, &b) ;`
- d. `echange(*a, *b) ;`

7. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
8. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
  - le contenu de la variable
9. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un opérateur du langage C
  - Un identificateur du langage C
  - Un mot-clef du langage C
10. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
11. Les instructions `i=0;`
- ```

while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 10 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 54
 - 58
 - 68
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`

15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur VRAI
 - d. n'a pas de valeur
16. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 45
17. `if (a%2 == 0) printf("bonjour") ;`
- a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Affiche bonjour quand *a* est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand *a* est un entier pair
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010101000
19. `printf("%c", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère 'A'
20. `if (a%2 = 0) printf("bonjour") ;`
- a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand *a* est un entier pair
 - d. Affiche bonjour quand *a* est un entier impair

Sujet n° 120

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
2. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère 'A'
3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 100
4. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
5. Après `char c ; c='a' ; c=c+1 ;`
 - a. *c* vaut 'a'
 - b. *c* vaut 'A'
 - c. *c* vaut 'b'
 - d. Un message d'erreur s'affiche
6. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand *a* est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand *a* est un entier pair
 - d. N'affiche rien (quelque soit la valeur de *a*)
7. L'adresse d'une variable *c* est :
 - a. le contenu de la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. ce vers quoi pointe la variable
 - d. l'adresse d'un pointeur sur la variable

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
9. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de `a`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
10. `printf("%c", 'A')`
- Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
12. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b);`
 - `echange(a, b);`
 - `echange(*a, *b);`
 - `echange(a++, b++);`
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
 - 58
 - 54
 - 62
15. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un opérateur du langage C
 - Un mot-clef du langage C

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010100110
- c. 111011010101000
- d. 111011010101111

17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

18. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier pair
- b. N'affiche rien (quelque soit la valeur de a)
- c. Affiche bonjour quand a est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

19. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 11 nombres
- c. vont boucler indéfiniment
- d. vont afficher 10 nombres

20. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur VRAI
- c. n'a pas de valeur
- d. a pour valeur FAUX

Sujet n° 121

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010101000
2. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. n'a pas de valeur
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur VRAI
4. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100
5. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Diminue de 1 la valeur de `a`
 - c. Utilise les opérateurs `<` et `=`
 - d. Réalise une affectation
6. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère `'A'`
7. `if (a%2 == 0) printf("bonjour") ;`
 - a. N'affiche rien (quelque soit la valeur de `a`)
 - b. Affiche bonjour quand `a` est un entier impair
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand `a` est un entier pair

8. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
9. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
 - N'affiche rien (quelque soit la valeur de *a*)
10. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*
11. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont afficher 11 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
12. `if` est :
- Un identificateur du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
13. L'adresse d'une variable c'est :
- le contenu de la variable
 - ce vers quoi pointe la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a, b);`
 - `echange(&a, &b);`
 - `echange(a++, b++);`
 - `echange(*a, *b);`
15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 68
 - 58
 - 54

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

17. Après `char c; c='a'; c=c+1;`

- a. `c` vaut `'A'`
- b. Un message d'erreur s'affiche
- c. `c` vaut `'b'`
- d. `c` vaut `'a'`

18. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère `'A'`

19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. ce vers quoi pointe le pointeur `a`
- c. la valeur de `a` tout simplement
- d. l'adresse de la variable `a`

20. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère `'A'`

Sujet n° 122

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un mot-clef du langage C
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C
2. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 54
 - b. 62
 - c. 68
 - d. 58
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101000
 - b. 111011010101111
 - c. 111011010100000
 - d. 111011010100110
5. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
6. Après `char c; c='a'; c=c+1;`
 - a. c vaut 'A'
 - b. c vaut 'b'
 - c. Un message d'erreur s'affiche
 - d. c vaut 'a'

7. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont afficher 10 nombres
b. vont afficher 11 nombres
c. vont afficher 9 nombres
d. vont boucler indéfiniment
8. L'expression `(0 == 7%3) || (1 == 9%3)`
a. n'a pas de valeur
b. a pour valeur VRAI
c. entraîne l'affichage d'un message d'erreur
d. a pour valeur FAUX
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
a. Permettent de d'affecter à `t[10]` la valeur 10
b. Permettent de d'affecter à `t[10]` la valeur 45
c. Permettent de d'affecter à `t[10]` la valeur 0
d. Permettent de d'affecter à `t[10]` la valeur 100
10. `printf("%c", 'A')`
a. Affiche le caractère 'A'
b. Affiche le code ASCII du caractère 'A'
c. Affiche le code ASCII du caractère stocké dans la variable `A`
d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
11. L'expression `a<=1`
a. Utilise les opérateurs `<` et `=`
b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
c. Diminue de 1 la valeur de `a`
d. Réalise une affectation
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
a. l'adresse de la variable `a`
b. ce vers quoi pointe le pointeur `a`
c. la valeur de `a` tout simplement
d. n'a pas de sens
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
a. ce vers quoi pointe le pointeur `a`
b. la valeur de `a` tout simplement
c. l'adresse de la variable `a`
d. n'a pas de sens
14. `printf("%i", 'A')`
a. Affiche le code ASCII du caractère 'A'
b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
c. Affiche le caractère 'A'
d. Affiche le code ASCII du caractère stocké dans la variable `A`

15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(a++, b++);`
- c. `echange(a, b);`
- d. `echange(*a, *b);`

16. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echage(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echage(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echage(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echage(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

18. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier pair
- d. Affiche bonjour quand `a` est un entier impair

19. L'adresse d'une variable c'est :

- a. le numéro de la case mémoire où le contenu de la variable est stocké
- b. l'adresse d'un pointeur sur la variable
- c. le contenu de la variable
- d. ce vers quoi pointe la variable

20. `if (a%2 = 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier pair
- d. Affiche bonjour quand `a` est un entier impair

Sujet n° 123

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le contenu de la variable
 - c. ce vers quoi pointe la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. la valeur de `a` tout simplement
3. Après `char c ; c='a' ; c=c+1 ;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut `'a'`
 - c. `c` vaut `'b'`
 - d. `c` vaut `'A'`
4. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. ce vers quoi pointe le pointeur `a`
6. L'expression `a<=1`
 - a. Réalise une affectation
 - b. Diminue de 1 la valeur de `a`
 - c. Utilise les opérateurs `<` et `=`
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(a, b) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(*a, *b) ;`

8. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont boucler indéfiniment
b. vont afficher 9 nombres
c. vont afficher 10 nombres
d. vont afficher 11 nombres
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
a. Permettent de d'affecter à `t[10]` la valeur 0
b. Permettent de d'affecter à `t[10]` la valeur 10
c. Permettent de d'affecter à `t[10]` la valeur 100
d. Permettent de d'affecter à `t[10]` la valeur 45
10. `printf("%c", 'A')`
a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
b. Affiche le code ASCII du caractère stocké dans la variable `A`
c. Affiche le code ASCII du caractère `'A'`
d. Affiche le caractère `'A'`
11. `if` est :
a. Une commande qu'on tape dans la fenêtre de commande
b. Un mot-clef du langage C
c. Un identificateur du langage C
d. Un opérateur du langage C
12. `printf("%c", A)`
a. Affiche le caractère `'A'`
b. Affiche le code ASCII du caractère `'A'`
c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
d. Affiche le code ASCII du caractère stocké dans la variable `A`
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
a. 111011010101111
b. 111011010101000
c. 111011010100000
d. 111011010100110
14. `if (a<5) printf("Bonjour") ; a=a+1 ;`
a. Augmente la valeur de `a` quelque soit `a`
b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
c. Affiche bonjour quelque soit `a`
d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
15. `printf("%i", 'A')`
a. Affiche le code ASCII du caractère `'A'`
b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
c. Affiche le caractère `'A'`
d. Affiche le code ASCII du caractère stocké dans la variable `A`

16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand a est un entier pair
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 68
 - b. 58
 - c. 62
 - d. 54
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. n'a pas de valeur
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur FAUX
 - d. a pour valeur VRAI
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
20. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de a)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier impair
 - d. Affiche bonjour quand a est un entier pair

Sujet n° 124

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a, b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(*a, *b) ;`

2. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

3. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

4. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un identificateur du langage C
- c. Un mot-clef du langage C
- d. Un opérateur du langage C

5. `if (a%2 == 0) printf("bonjour") ;`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Affiche bonjour quand `a` est un entier pair
- d. Affiche bonjour quand `a` est un entier impair

6. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. l'adresse de la variable `a`
- c. ce vers quoi pointe le pointeur `a`
- d. n'a pas de sens

7. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- c. Affiche bonjour quelque soit `a`
- d. Augmente la valeur de `a` quelque soit `a`

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
11. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100000
  - 111011010101000
  - 111011010100110
13. L'adresse d'une variable c'est :
- le contenu de la variable
  - ce vers quoi pointe la variable
  - l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
  - a pour valeur VRAI
15. `printf("%c", 'A')`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`

16. L'expression `a<=1`
- a. Diminue de 1 la valeur de  $a$
  - b. Réalise une affectation
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Utilise les opérateurs `<` et `=`
17. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
18. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
  - b.  $c$  vaut `'b'`
  - c.  $c$  vaut `'A'`
  - d.  $c$  vaut `'a'`
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 58
  - b. 68
  - c. 54
  - d. 62
20. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`

# Sujet n° 125

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'A'`
  - d. `c` vaut `'b'`
2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - b. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - c. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - d. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier impair
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de `a`)
4. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 54
  - c. 68
  - d. 58
5. `printf("%i", 'A')`
  - a. Affiche le caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
6. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. l'adresse de la variable `a`
  - c. ce vers quoi pointe le pointeur `a`
  - d. la valeur de `a` tout simplement

8. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur VRAI
  - b. a pour valeur FAUX
  - c. n'a pas de valeur
  - d. entraîne l'affichage d'un message d'erreur
9. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le contenu de la variable
10. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable A
11. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand a est un entier pair
  - c. N'affiche rien (quelque soit la valeur de a)
  - d. Affiche bonjour quand a est un entier impair
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(&a, &b);`
  - b. `echange(a, b);`
  - c. `echange(*a, *b);`
  - d. `echange(a++, b++);`
13. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Utilise les opérateurs `<` et `=`
  - c. Réalise une affectation
  - d. Diminue de 1 la valeur de a
14. `if` est :
  - a. Un opérateur du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un mot-clef du langage C
15. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 10



16. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
  - b. 111011010100000
  - c. 111011010101000
  - d. 111011010101111
18. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. ce vers quoi pointe le pointeur *a*
  - b. la valeur de *a* tout simplement
  - c. l'adresse de la variable *a*
  - d. n'a pas de sens
19. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont afficher 9 nombres
 - b. vont afficher 10 nombres
 - c. vont afficher 11 nombres
 - d. vont boucler indéfiniment
20. `if (a<5) printf("Bonjour"); a=a+1 ;`
- a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - b. Affiche bonjour quelque soit *a*
 - c. Augmente la valeur de *a* quelque soit *a*
 - d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*

Sujet n° 126

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur FAUX
 - c. a pour valeur VRAI
 - d. n'a pas de valeur
2. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`
3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(a, b) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(a++, b++) ;`
4. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère `'A'`
5. L'expression `a<=1`
 - a. Réalise une affectation
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Diminue de 1 la valeur de `a`
 - d. Utilise les opérateurs `<` et `=`
6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. n'a pas de sens
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`

7. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont afficher 10 nombres
b. vont afficher 9 nombres
c. vont afficher 11 nombres
d. vont boucler indéfiniment
8. `if (a%2 == 0) printf("bonjour") ;`
a. Affiche bonjour quand a est un entier impair
b. N'affiche rien (quelque soit la valeur de a)
c. Affiche bonjour quand a est un entier pair
d. Déclenche le message d'erreur `invalid lvalue in assignment`
9. `printf("%i", 'A')`
a. Affiche le caractère dont le code ASCII est stocké dans la variable A
b. Affiche le code ASCII du caractère 'A'
c. Affiche le caractère 'A'
d. Affiche le code ASCII du caractère stocké dans la variable A
10. L'adresse d'une variable c'est :
a. l'adresse d'un pointeur sur la variable
b. ce vers quoi pointe la variable
c. le numéro de la case mémoire où le contenu de la variable est stocké
d. le contenu de la variable
11. `if` est :
a. Une commande qu'on tape dans la fenêtre de commande
b. Un opérateur du langage C
c. Un mot-clef du langage C
d. Un identificateur du langage C
12. Après `char c ; c='a' ; c=c+1 ;`
a. c vaut 'A'
b. c vaut 'b'
c. c vaut 'a'
d. Un message d'erreur s'affiche
13. `if (a%2 == 0) printf("bonjour") ;`
a. Affiche bonjour quand a est un entier pair
b. Affiche bonjour quand a est un entier impair
c. N'affiche rien (quelque soit la valeur de a)
d. Déclenche le message d'erreur `invalid lvalue in assignment`
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
a. 62
b. 68
c. 58
d. 54

15. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
17. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010101000
 - 111011010100110
 - 111011010101111
19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à $t[10]$ la valeur 100
 - Permettent de d'affecter à $t[10]$ la valeur 45
 - Permettent de d'affecter à $t[10]$ la valeur 0
 - Permettent de d'affecter à $t[10]$ la valeur 10
20. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'

Sujet n° 127

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère 'A'
2. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Affiche bonjour quand *a* est un entier impair
 - d. Affiche bonjour quand *a* est un entier pair
3. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère 'A'
4. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit *a*
 - b. Augmente la valeur de *a* quelque soit *a*
 - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 54
 - c. 58
 - d. 68
6. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. a pour valeur FAUX
 - c. entraîne l'affichage d'un message d'erreur
 - d. n'a pas de valeur
7. L'expression `a<=1`
 - a. Réalise une affectation
 - b. Diminue de 1 la valeur de *a*
 - c. Utilise les opérateurs `<` et `=`
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon

8. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 9 nombres
 - vont afficher 10 nombres
 - vont afficher 11 nombres
 - vont boucler indéfiniment
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
10. `if` est :
- Un opérateur du langage C
 - Un mot-clef du langage C
 - Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
11. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier pair
12. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
13. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
15. `printf("%i", 'A')`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`

16. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'b'`
 - b. `c` vaut `'a'`
 - c. `c` vaut `'A'`
 - d. Un message d'erreur s'affiche
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - b. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - c. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010101111
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(*a, *b) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(a, b) ;`
20. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`

Sujet n° 128

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un mot-clef du langage C
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C
2. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand a est un entier pair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Affiche bonjour quand a est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 62
 - c. 68
 - d. 54
4. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 100
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 0
5. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
6. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
7. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. Diminue de 1 la valeur de a
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Réalise une affectation

8. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de a quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
 - `echange(*a, *b) ;`
10. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
12. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - a pour valeur FAUX
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
14. Les instructions
- ```
i=0;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
15. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
  - `c` vaut 'A'
  - `c` vaut 'b'
  - `c` vaut 'a'

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
  - b. 111011010100000
  - c. 111011010101111
  - d. 111011010100110
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. la valeur de `a` tout simplement
  - c. n'a pas de sens
  - d. ce vers quoi pointe le pointeur `a`
18. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. le contenu de la variable
  - c. ce vers quoi pointe la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
19. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
20. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Affiche bonjour quand `a` est un entier pair



# Sujet n° 129

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. l'adresse d'un pointeur sur la variable
  - d. ce vers quoi pointe la variable
2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
3. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 10
4. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier impair
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier pair
5. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur VRAI
  - c. n'a pas de valeur
  - d. a pour valeur FAUX
6. L'expression `a<=1`
  - a. Diminue de 1 la valeur de `a`
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Réalise une affectation
7. `if (a%2 = 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier pair
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier impair

8. `printf("%c", A)`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a++, b++) ;`
  - `echange(*a, *b) ;`
  - `echange(a, b) ;`
  - `echange(&a, &b) ;`
10. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010101000
  - 111011010101111
  - 111011010100000
12. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
  - l'adresse de la variable *a*
  - n'a pas de sens
  - la valeur de *a* tout simplement
13. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de *a* tout simplement
  - n'a pas de sens
  - l'adresse de la variable *a*
  - ce vers quoi pointe le pointeur *a*
14. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
  - Un mot-clef du langage C
  - Un opérateur du langage C
15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 68
  - 58

16. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
17. Après `char c ; c='a' ; c=c+1 ;`
- a. *c* vaut 'A'
  - b. *c* vaut 'b'
  - c. Un message d'erreur s'affiche
  - d. *c* vaut 'a'
18. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- a. vont boucler indéfiniment
  - b. vont afficher 11 nombres
  - c. vont afficher 9 nombres
  - d. vont afficher 10 nombres
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le caractère 'A'
20. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Augmente la valeur de *a* quelque soit *a*
  - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - c. Affiche bonjour quelque soit *a*
  - d. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*

# Sujet n° 130

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100110
- c. 111011010100000
- d. 111011010101000

2. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 45

3. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère `'A'`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

4. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Augmente la valeur de `a` quelque soit `a`
- b. Affiche bonjour quelque soit `a`
- c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`

5. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```

- a. vont afficher 11 nombres
- b. vont afficher 9 nombres
- c. vont boucler indéfiniment
- d. vont afficher 10 nombres

6. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère `'A'`



7. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b);`
  - `echange(a, b);`
  - `echange(a++, b++);`
  - `echange(*a, *b);`
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 68
  - 58
  - 62
11. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier pair
  - Affiche bonjour quand a est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de `a`)
12. Après `char c; c='a'; c=c+1;`
- `c` vaut 'A'
  - `c` vaut 'b'
  - Un message d'erreur s'affiche
  - `c` vaut 'a'
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - n'a pas de valeur
  - a pour valeur FAUX
  - a pour valeur VRAI

15. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. le contenu de la variable
  - c. ce vers quoi pointe la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
  - b. l'adresse de la variable `a`
  - c. la valeur de `a` tout simplement
  - d. n'a pas de sens
18. `if` est :
- a. Un opérateur du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un mot-clef du langage C
19. L'expression `a<=1`
- a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Diminue de 1 la valeur de  $a$
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
20. `printf("%c", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`

# Sujet n° 131

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 10 nombres
- c. vont boucler indéfiniment
- d. vont afficher 11 nombres

2. `printf("%i", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- d. Affiche le code ASCII du caractère stocké dans la variable *A*

3. `printf("%c", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

4. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 54
- c. 58
- d. 62

5. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100000
- c. 111011010100110
- d. 111011010101000

6. `if (a%2 == 0) printf("bonjour") ;`

- a. N'affiche rien (quelque soit la valeur de *a*)
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand *a* est un entier impair
- d. Affiche bonjour quand *a* est un entier pair

7. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
8. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
  - n'a pas de valeur
  - a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur
9. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère 'A'
10. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
  - N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier pair
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a++, b++) ;`
  - `echange(a, b) ;`
  - `echange(&a, &b) ;`
  - `echange(*a, *b) ;`
12. L'expression `a<=1`
- Diminue de 1 la valeur de *a*
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
  - Réalise une affectation
13. Après `char c ; c='a' ; c=c+1 ;`
- c* vaut 'A'
  - c* vaut 'a'
  - Un message d'erreur s'affiche
  - c* vaut 'b'
14. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de *a* tout simplement
  - l'adresse de la variable *a*
  - n'a pas de sens
  - ce vers quoi pointe le pointeur *a*

15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

16. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit *a*
- b. Augmente la valeur de *a* quelque soit *a*
- c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
- d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*

17. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un mot-clef du langage C
- c. Un opérateur du langage C
- d. Un identificateur du langage C

18. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. ce vers quoi pointe la variable
- c. le contenu de la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 45

20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. l'adresse de la variable `a`
- d. ce vers quoi pointe le pointeur `a`

# Sujet n° 132

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.



## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

2. `printf("%c", 'A')`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

3. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable A
- b. Affiche le caractère dont le code ASCII est stocké dans la variable A
- c. Affiche le caractère 'A'
- d. Affiche le code ASCII du caractère 'A'

4. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de *a*)
- c. Affiche bonjour quand *a* est un entier impair
- d. Affiche bonjour quand *a* est un entier pair

5. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable A
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le caractère 'A'

6. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur VRAI
- b. n'a pas de valeur
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur FAUX

7. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
- b. Affiche le code ASCII du caractère stocké dans la variable A
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le caractère 'A'

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
9. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
10. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
  - `c` vaut `'A'`
  - Un message d'erreur s'affiche
  - `c` vaut `'a'`
11. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont boucler indéfiniment
  - vont afficher 10 nombres
  - vont afficher 11 nombres
  - vont afficher 9 nombres
12. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 68
  - 54
  - 58
14. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
15. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un mot-clef du langage C
  - Un identificateur du langage C
  - Un opérateur du langage C

16. L'adresse d'une variable c'est :
- a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(&a, &b) ;`
  - b. `echange(a, b) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(a++, b++) ;`
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
  - b. 111011010100000
  - c. 111011010101000
  - d. 111011010101111
19. L'expression `a<=1`
- a. Utilise les opérateurs `<` et `=`
  - b. Diminue de 1 la valeur de `a`
  - c. Réalise une affectation
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
20. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 100
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 45

# Sujet n° 133

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 100
2. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand `a` est un entier pair
  - d. N'affiche rien (quelque soit la valeur de `a`)
3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(*a, *b) ;`
  - b. `echange(&a, &b) ;`
  - c. `echange(a++, b++) ;`
  - d. `echange(a, b) ;`
4. `if` est :
  - a. Un mot-clef du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un opérateur du langage C
5. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère '`A`'
  - b. Affiche le caractère '`A`'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
6. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable
7. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 58
  - c. 54
  - d. 68

8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

9. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Réalise une affectation
- c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- d. Diminue de 1 la valeur de  $a$

10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. n'a pas de sens
- c. la valeur de `a` tout simplement
- d. l'adresse de la variable `a`

11. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur VRAI
- b. a pour valeur FAUX
- c. n'a pas de valeur
- d. entraîne l'affichage d'un message d'erreur

12. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de  $a$ )
- b. Affiche bonjour quand  $a$  est un entier impair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand  $a$  est un entier pair

13. `printf("%c", A)`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

15. Les instructions `i=0;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 11 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment



16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
  - b. 111011010100110
  - c. 111011010101111
  - d. 111011010100000
17. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour quelque soit  $a$
  - b. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Augmente la valeur de  $a$  quelque soit  $a$
18. `printf("%c", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
19. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
20. Après `char c; c='a'; c=c+1;`
- a.  $c$  vaut 'A'
  - b. Un message d'erreur s'affiche
  - c.  $c$  vaut 'a'
  - d.  $c$  vaut 'b'

# Sujet n° 134

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur FAUX
  - c. n'a pas de valeur
  - d. a pour valeur VRAI
2. Les instructions 

```
i=0 ;
while(i<10)
 printf("%i ", i);
 i++ ;
```

  - a. vont boucler indéfiniment
  - b. vont afficher 10 nombres
  - c. vont afficher 11 nombres
  - d. vont afficher 9 nombres
3. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Augmente la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
4. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. ce vers quoi pointe la variable
  - c. le contenu de la variable
  - d. l'adresse d'un pointeur sur la variable
5. L'expression `a<=1`
  - a. Réalise une affectation
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Diminue de 1 la valeur de  $a$
6. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Affiche bonjour quand  $a$  est un entier pair

7. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(&a, &b) ;`
  - `echange(a, b) ;`
  - `echange(*a, *b) ;`
  - `echange(a++, b++) ;`
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100000
  - 111011010100110
  - 111011010101000
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
12. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
13. Après `char c ; c='a' ; c=c+1 ;`
- c* vaut 'b'
  - Un message d'erreur s'affiche
  - c* vaut 'a'
  - c* vaut 'A'
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 54
  - 58
  - 68

15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. n'a pas de sens
16. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère `'A'`
17. `if` est :
- a. Un mot-clef du langage C
  - b. Une commande qu'on tape dans la fenêtre de commande
  - c. Un opérateur du langage C
  - d. Un identificateur du langage C
18. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`
19. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand `a` est un entier impair
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de `a`)
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
  - b. l'adresse de la variable `a`
  - c. la valeur de `a` tout simplement
  - d. ce vers quoi pointe le pointeur `a`

# Sujet n° 135

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 10 nombres
- b. vont boucler indéfiniment
- c. vont afficher 9 nombres
- d. vont afficher 11 nombres

2. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. ce vers quoi pointe la variable
- c. le numéro de la case mémoire où le contenu de la variable est stocké
- d. le contenu de la variable

3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b);`
- b. `echange(a, b);`
- c. `echange(a++, b++);`
- d. `echange(*a, *b);`

4. `if` est :

- a. Un opérateur du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un identificateur du langage C
- d. Un mot-clef du langage C

5. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit *a*
- b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
- c. Augmente la valeur de *a* quelque soit *a*
- d. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*

6. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010100000
- c. 111011010101000
- d. 111011010101111

7. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut 'b'
  - `c` vaut 'A'
  - Un message d'erreur s'affiche
  - `c` vaut 'a'
8. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier pair
  - Affiche bonjour quand `a` est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens
10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
12. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère 'A'
13. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 68
  - 62
  - 58

15. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier impair
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
16. L'expression `a<=1`
- a. Diminue de 1 la valeur de  $a$
  - b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - c. Utilise les opérateurs `<` et `=`
  - d. Réalise une affectation
17. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 100
  - d. Permettent de d'affecter à `t[10]` la valeur 0
18. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère `'A'`
20. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
  - b. a pour valeur FAUX
  - c. n'a pas de valeur
  - d. entraîne l'affichage d'un message d'erreur

# Sujet n° 136

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b) ;`
- b. `echange(&a, &b) ;`
- c. `echange(a, b) ;`
- d. `echange(a++, b++) ;`

2. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. entraîne l'affichage d'un message d'erreur
- c. a pour valeur FAUX
- d. a pour valeur VRAI

3. `printf("%i", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère 'A'

4. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 54
- b. 62
- c. 58
- d. 68

5. `if` est :

- a. Un mot-clef du langage C
- b. Un opérateur du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un identificateur du langage C

6. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 11 nombres

7. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'b'`
  - c. `c` vaut `'a'`
  - d. `c` vaut `'A'`
8. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. le contenu de la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
9. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`
10. `printf("%c", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010101111
  - c. 111011010101000
  - d. 111011010100110
12. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier impair
  - c. Affiche bonjour quand `a` est un entier pair
  - d. N'affiche rien (quelque soit la valeur de `a`)
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - d. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`

15. `if (a<5) printf("Bonjour"); a=a+1;`
- N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
16. L'expression `a<=1`
- A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Diminue de 1 la valeur de  $a$
  - Réalise une affectation
  - Utilise les opérateurs `<` et `=`
17. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier impair
  - Affiche bonjour quand  $a$  est un entier pair
18. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère `'A'`
19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`



# Sujet n° 137

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère 'A'
2. `if` est :
  - a. Un identificateur du langage C
  - b. Un mot-clef du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un opérateur du langage C
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. N'affiche rien (quelque soit la valeur de  $a$ )
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
4. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère 'A'
5. `printf("%c", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
6. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Réalise une affectation
  - c. Utilise les opérateurs `<` et `=`
  - d. Diminue de 1 la valeur de  $a$
7. Les instructions 

```
i=0 ;
while(i<10)
 printf("%i ", i) ;
i++ ;
```

  - a. vont afficher 10 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 9 nombres
  - d. vont afficher 11 nombres

8. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - le contenu de la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 68
  - 62
  - 54
10. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut 'A'
  - `c` vaut 'a'
  - `c` vaut 'b'
  - Un message d'erreur s'affiche
11. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable A
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le caractère 'A'
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Augmente la valeur de *a* quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour quelque soit *a*
13. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a++, b++) ;`
  - `echange(&a, &b) ;`
  - `echange(a, b) ;`
  - `echange(*a, *b) ;`
15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45

16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
  - b. n'a pas de valeur
  - c. a pour valeur FAUX
  - d. a pour valeur VRAI
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
19. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. la valeur de `a` tout simplement
  - c. ce vers quoi pointe le pointeur `a`
  - d. n'a pas de sens
20. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :
- a. `111011010100000`
  - b. `111011010101111`
  - c. `111011010101000`
  - d. `111011010100110`

# Sujet n° 138

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
2. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le caractère `'A'`
3. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 10
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a++, b++) ;`
  - b. `echange(&a, &b) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(a, b) ;`
5. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. l'adresse d'un pointeur sur la variable
  - c. ce vers quoi pointe la variable
  - d. le contenu de la variable
6. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère `'A'`



7. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010101000
  - 111011010100000
  - 111011010101111
8. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - `c` vaut `'a'`
  - `c` vaut `'b'`
  - `c` vaut `'A'`
9. `if` est :
- Un mot-clef du langage C
  - Un opérateur du langage C
  - Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
  - 54
  - 58
  - 62
11. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
13. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- `a` pour valeur VRAI
  - n'a pas de valeur
  - entraîne l'affichage d'un message d'erreur
  - `a` pour valeur FAUX

15. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
16. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
18. Les instructions
- ```
i=0;
while(i<10)
    printf("%i ", i);
    i++;
```
- vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont afficher 9 nombres
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
20. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Diminue de 1 la valeur de a

Sujet n° 139

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101000
- c. 111011010100000
- d. 111011010101111

2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b);`
- b. `echange(*a, *b);`
- c. `echange(&a, &b);`
- d. `echange(a++, b++);`

4. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère `'A'`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

5. `if` est :

- a. Un opérateur du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un mot-clef du langage C
- d. Un identificateur du langage C

6. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur FAUX
- b. a pour valeur VRAI
- c. n'a pas de valeur
- d. entraîne l'affichage d'un message d'erreur

7. `printf("%c", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
 - 54
 - 58
 - 62
9. Après `char c ; c='a' ; c=c+1 ;`
- c* vaut 'a'
 - c* vaut 'b'
 - Un message d'erreur s'affiche
 - c* vaut 'A'
10. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand *a* est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
11. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable *a*
 - la valeur de *a* tout simplement
 - ce vers quoi pointe le pointeur *a*
12. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à *t*[10] la valeur 45
 - Permettent de d'affecter à *t*[10] la valeur 10
 - Permettent de d'affecter à *t*[10] la valeur 0
 - Permettent de d'affecter à *t*[10] la valeur 100
13. `if (a%2 = 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier pair
 - Affiche bonjour quand *a* est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
14. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
15. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*

16. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
 - vont afficher 11 nombres
17. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
19. L'adresse d'une variable c'est :
- le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
20. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Diminue de 1 la valeur de `a`
 - Utilise les opérateurs `<` et `=`
 - Réalise une affectation

Sujet n° 140

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Augmente la valeur de a quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
2. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
3. On suppose que a a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de a tout simplement
 - b. ce vers quoi pointe le pointeur a
 - c. l'adresse de la variable a
 - d. n'a pas de sens
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101000
5. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 0
6. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. ce vers quoi pointe la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
7. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A

8. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
9. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
 - `c` vaut `'a'`
 - Un message d'erreur s'affiche
 - `c` vaut `'A'`
10. L'expression `a<=1`
- Réalise une affectation
 - Diminue de 1 la valeur de `a`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
 - `echange(a++, b++) ;`
12. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
 - Un identificateur du langage C
 - Un mot-clef du langage C
13. `printf("%c", 'A')`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI

16. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier pair
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
18. Les instructions
- ```

i=0;
while(i<10)
 printf("%i ", i);
 i++;

```
- vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
  - vont afficher 11 nombres
19. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand  $a$  est un entier impair
  - Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de  $a$ )
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 68
  - 62
  - 58

# Sujet n° 141

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

2. Les instructions `i=0;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 10 nombres
- c. vont afficher 11 nombres
- d. vont boucler indéfiniment

3. Après `char c; c='a'; c=c+1;`

- a. `c` vaut `'a'`
- b. `c` vaut `'b'`
- c. `c` vaut `'A'`
- d. Un message d'erreur s'affiche

4. L'adresse d'une variable c'est :

- a. le contenu de la variable
- b. l'adresse d'un pointeur sur la variable
- c. ce vers quoi pointe la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

5. L'expression `a<=1`

- a. Réalise une affectation
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Utilise les opérateurs `<` et `=`
- d. Diminue de 1 la valeur de `a`

6. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 62
- c. 58
- d. 54

7. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
8. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable *a*
  - ce vers quoi pointe le pointeur *a*
  - la valeur de *a* tout simplement
  - n'a pas de sens
9. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
10. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
  - la valeur de *a* tout simplement
  - n'a pas de sens
  - l'adresse de la variable *a*
11. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier pair
12. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
13. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
  - a pour valeur VRAI
  - a pour valeur FAUX
  - n'a pas de valeur
15. `if (a%2 = 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
  - N'affiche rien (quelque soit la valeur de *a*)



16. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
17. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour quelque soit *a*
  - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
18. En cours, on a vu comment à l'aide de pointeurs définir une fonction **echange** qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables **a** et **b** on doit écrire :
- `echange(a, b) ;`
  - `echange(a++, b++) ;`
  - `echange(*a, *b) ;`
  - `echange(&a, &b) ;`
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101111
  - 111011010101000
  - 111011010100110
20. `if` est :
- Un opérateur du langage C
  - Un identificateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande

# Sujet n° 142

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 54
  - b. 58
  - c. 68
  - d. 62
2. `printf("%c", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101111
  - b. 111011010101000
  - c. 111011010100000
  - d. 111011010100110
4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. l'adresse de la variable `a`
  - c. ce vers quoi pointe le pointeur `a`
  - d. la valeur de `a` tout simplement
5. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand *a* est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de *a*)
  - d. Affiche bonjour quand *a* est un entier impair
6. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(*a, *b) ;`
- d. `echange(&a, &b) ;`

8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 45
- d. Permettent de d'affecter à `t[10]` la valeur 100

9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
- b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
- c. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
- d. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`

10. `if` est :

- a. Un mot-clef du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un opérateur du langage C
- d. Un identificateur du langage C

11. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Augmente la valeur de `a` quelque soit `a`
- b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- d. Affiche bonjour quelque soit `a`

12. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut `'A'`
- b. `c` vaut `'a'`
- c. `c` vaut `'b'`
- d. Un message d'erreur s'affiche

13. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

14. L'expression `a<=1`

- a. Diminue de 1 la valeur de `a`
- b. Réalise une affectation
- c. Utilise les opérateurs `<` et `=`
- d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon

15. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. Affiche bonjour quand  $a$  est un entier impair
16. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$
17. Les instructions `i=0;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont afficher 9 nombres
 - b. vont afficher 11 nombres
 - c. vont boucler indéfiniment
 - d. vont afficher 10 nombres
18. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. a pour valeur VRAI
20. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable a
 - b. n'a pas de sens
 - c. ce vers quoi pointe le pointeur a
 - d. la valeur de a tout simplement

Sujet n° 143

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. n'a pas de sens
- c. l'adresse de la variable `a`
- d. la valeur de `a` tout simplement

2. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. N'affiche rien (quelque soit la valeur de `a`)
- d. Affiche bonjour quand `a` est un entier pair

3. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 11 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment

4. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 10
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 100

5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

6. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit `a`
- b. Augmente la valeur de `a` quelque soit `a`
- c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`

7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(a, b) ;`
 - `echange(*a, *b) ;`
 - `echange(&a, &b) ;`
9. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
 - `c` vaut `'A'`
 - Un message d'erreur s'affiche
 - `c` vaut `'b'`
10. `printf("%i", A)`
- Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010100110
 - 111011010100000
 - 111011010101111
12. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
 - Affiche bonjour quand `a` est un entier pair
13. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C
 - Un opérateur du langage C
14. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable

15. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. n'a pas de valeur
 - b. a pour valeur FAUX
 - c. a pour valeur VRAI
 - d. entraîne l'affichage d'un message d'erreur
17. `printf("%c", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
18. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
 - b. 54
 - c. 68
 - d. 58
20. L'expression `a<=1`
- a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de *a*

Sujet n° 144

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101111
- c. 111011010100110
- d. 111011010101000

2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. la valeur de `a` tout simplement
- b. l'adresse de la variable `a`
- c. ce vers quoi pointe le pointeur `a`
- d. n'a pas de sens

3. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère `'A'`

4. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

5. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur VRAI
- b. entraîne l'affichage d'un message d'erreur
- c. a pour valeur FAUX
- d. n'a pas de valeur

6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a++, b++) ;`
- c. `echange(a, b) ;`
- d. `echange(*a, *b) ;`

7. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
9. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
10. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le contenu de la variable
11. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
- vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
12. `if (a%2 = 0) printf("bonjour") ;`
- Affiche bonjour quand *a* est un entier pair
 - Affiche bonjour quand *a* est un entier impair
 - N'affiche rien (quelque soit la valeur de *a*)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
13. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
14. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C

15. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 54
 - 68
 - 62
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
18. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
19. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
 - `c` vaut `'a'`
 - `c` vaut `'b'`
 - `c` vaut `'A'`
20. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de `a`
 - Réalise une affectation
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon

Sujet n° 145

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
2. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable *a*
 - c. la valeur de *a* tout simplement
 - d. ce vers quoi pointe le pointeur *a*
3. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier impair
4. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. le contenu de la variable
5. L'expression `a--`
 - a. Réalise une affectation
 - b. Diminue de 1 la valeur de *a*
 - c. Utilise les opérateurs `<` et `=`
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
6. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
7. `if` est :
 - a. Un opérateur du langage C
 - b. Un mot-clef du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un identificateur du langage C

8. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur FAUX
 - a pour valeur VRAI
10. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
11. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
 - l'adresse de la variable *a*
 - la valeur de *a* tout simplement
 - n'a pas de sens
12. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
13. `printf("%c", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
14. Les instructions
- ```

i=0;
while(i<10)
 printf("%i ", i);
 i++;

```
- vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
15. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010100000
  - 111011010101111
  - 111011010101000

16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 54
  - b. 68
  - c. 62
  - d. 58
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(&a, &b);`
  - b. `echange(*a, *b);`
  - c. `echange(a++, b++);`
  - d. `echange(a, b);`
18. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'b'`
  - d. `c` vaut `'A'`
19. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand `a` est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier impair
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

# Sujet n° 146

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier pair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier impair
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010101000
  - b. 111011010100110
  - c. 111011010101111
  - d. 111011010100000
3. `printf("%c", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
4. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Augmente la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
5. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable
6. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
7. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'



8. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 100
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 68
  - 58
  - 54
10. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
  - `c` vaut `'A'`
  - Un message d'erreur s'affiche
  - `c` vaut `'a'`
11. `printf("%i", A)`
- Affiche le caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
12. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
13. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont boucler indéfiniment
  - vont afficher 11 nombres
  - vont afficher 10 nombres
  - vont afficher 9 nombres
14. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
  - `echange(*a, *b) ;`
  - `echange(a, b) ;`
  - `echange(a++, b++) ;`

16. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le code ASCII du caractère stocké dans la variable A
17. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur FAUX
  - a pour valeur VRAI
  - entraîne l'affichage d'un message d'erreur
18. `if` est :
- Un opérateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
20. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`

# Sujet n° 147

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Affiche bonjour quand `a` est un entier impair
  - d. N'affiche rien (quelque soit la valeur de `a`)
2. `printf("%i", 'A')`
  - a. Affiche le caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de `a` tout simplement
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
4. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
5. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. Augmente la valeur de `a` quelque soit `a`
  - b. Affiche bonjour quelque soit `a`
  - c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
6. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le caractère `'A'`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
7. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 0

8. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

9. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

10. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. l'adresse de la variable *a*
- c. la valeur de *a* tout simplement
- d. ce vers quoi pointe le pointeur *a*

11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :

- a. `echange(a, b);`
- b. `echange(a++, b++);`
- c. `echange(*a, *b);`
- d. `echange(&a, &b);`

12. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Réalise une affectation
- c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- d. Diminue de 1 la valeur de *a*

13. `printf("%i", A)`

- a. Affiche le caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

14. Après `char c; c='a'; c=c+1;`

- a. *c* vaut 'a'
- b. Un message d'erreur s'affiche
- c. *c* vaut 'A'
- d. *c* vaut 'b'

15. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010100110
- c. 111011010101111
- d. 111011010101000

16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 54
  - b. 68
  - c. 62
  - d. 58
17. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```
- a. vont afficher 11 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 10 nombres
 - d. vont afficher 9 nombres
18. `if` est :
- a. Un mot-clef du langage C
 - b. Un opérateur du langage C
 - c. Un identificateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
 - b. a pour valeur FAUX
 - c. entraîne l'affichage d'un message d'erreur
 - d. n'a pas de valeur
20. `if (a%2 = 0) printf("bonjour") ;`
- a. N'affiche rien (quelque soit la valeur de *a*)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand *a* est un entier pair
 - d. Affiche bonjour quand *a* est un entier impair

Sujet n° 148

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101000
- c. 111011010101111
- d. 111011010100000

2. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le caractère 'A'

3. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand *a* est un entier impair
- c. Affiche bonjour quand *a* est un entier pair
- d. N'affiche rien (quelque soit la valeur de *a*)

4. `printf("%c", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

5. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 58
- c. 62
- d. 54

6. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur *a*
- b. l'adresse de la variable *a*
- c. la valeur de *a* tout simplement
- d. n'a pas de sens

7. `if (a%2 = 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de *a*)
- b. Affiche bonjour quand *a* est un entier pair
- c. Affiche bonjour quand *a* est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

8. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
9. `if` est :
- Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un opérateur du langage C
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
 - `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
11. `printf("%c", 'A')`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
13. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
14. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
15. L'adresse d'une variable c'est :
- le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

17. L'expression `a<=1`

- a. Diminue de 1 la valeur de a
- b. Réalise une affectation
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Utilise les opérateurs `<` et `=`

18. Après `char c; c='a'; c=c+1;`

- a. c vaut 'b'
- b. Un message d'erreur s'affiche
- c. c vaut 'A'
- d. c vaut 'a'

19. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont afficher 10 nombres
- c. vont afficher 9 nombres
- d. vont boucler indéfiniment

20. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 0
- c. Permettent de d'affecter à `t[10]` la valeur 45
- d. Permettent de d'affecter à `t[10]` la valeur 10

Sujet n° 149

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour quelque soit a
 - b. Affiche bonjour et augmente la valeur de a quelque soit a
 - c. Augmente la valeur de a quelque soit a
 - d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
2. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. a pour valeur FAUX
 - c. n'a pas de valeur
 - d. entraîne l'affichage d'un message d'erreur
3. `if` est :
 - a. Un mot-clef du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un opérateur du langage C
 - d. Un identificateur du langage C
4. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(*a, *b) ;`
 - b. `echange(a, b) ;`
 - c. `echange(&a, &b) ;`
 - d. `echange(a++, b++) ;`
6. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 10
 - d. Permettent de d'affecter à `t[10]` la valeur 100
7. `if (a%2 == 0) printf("bonjour") ;`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Affiche bonjour quand a est un entier pair
 - c. Affiche bonjour quand a est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`

8. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - N'affiche rien (quelque soit la valeur de a)
 - Affiche bonjour quand a est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
10. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont boucler indéfiniment
 - vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont afficher 10 nombres
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 54
 - 68
 - 62
12. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
 - 111011010101000
 - 111011010101111
 - 111011010100000
14. L'adresse d'une variable c'est :
- le contenu de la variable
 - ce vers quoi pointe la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
15. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de a
 - Réalise une affectation

16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
17. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
18. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
 - `c` vaut `'A'`
 - `c` vaut `'a'`
 - Un message d'erreur s'affiche
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`

Sujet n° 150

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 9 nombres
- b. vont afficher 10 nombres
- c. vont afficher 11 nombres
- d. vont boucler indéfiniment

2. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour et augmente la valeur de a quelque soit a
- b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- c. Affiche bonjour quelque soit a
- d. Augmente la valeur de a quelque soit a

3. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. Affiche bonjour quand a est un entier pair
- c. N'affiche rien (quelque soit la valeur de a)
- d. Affiche bonjour quand a est un entier impair

4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101000
- c. 111011010101111
- d. 111011010100110

5. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable A
- b. Affiche le code ASCII du caractère 'A'
- c. Affiche le caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

6. `if (a%2 = 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier impair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. N'affiche rien (quelque soit la valeur de a)
- d. Affiche bonjour quand a est un entier pair

7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
8. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
 - `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
10. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
 - Un message d'erreur s'affiche
 - `c` vaut `'A'`
 - `c` vaut `'b'`
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
12. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable
 - ce vers quoi pointe la variable
13. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
14. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
 - Réalise une affectation
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`

15. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - a pour valeur FAUX
 - a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
17. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 68
 - 54
18. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
19. `if` est :
- Un mot-clef du langage C
 - Un identificateur du langage C
 - Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
20. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A

Sujet n° 151

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère 'A'
2. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 45
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur VRAI
 - d. a pour valeur FAUX
4. L'adresse d'une variable c'est :
 - a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. le contenu de la variable
 - c. ce vers quoi pointe la variable
 - d. l'adresse d'un pointeur sur la variable
5. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. ce vers quoi pointe le pointeur *a*
 - b. l'adresse de la variable *a*
 - c. la valeur de *a* tout simplement
 - d. n'a pas de sens
6. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Augmente la valeur de *a* quelque soit *a*
 - b. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - c. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - d. Affiche bonjour quelque soit *a*
7. `if` est :
 - a. Un mot-clef du langage C
 - b. Un opérateur du langage C
 - c. Un identificateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
9. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)
10. `printf("%c", A)`
- Affiche le code ASCII du caractère '`A`'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère '`A`'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
11. Les instructions `i=0;`
- ```
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 54
  - 68
  - 58
13. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
  - Utilise les opérateurs `<` et `=`
  - Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
14. Après `char c; c='a'; c=c+1;`
- `c` vaut '`a`'
  - Un message d'erreur s'affiche
  - `c` vaut '`A`'
  - `c` vaut '`b`'
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++);`
  - `echange(a, b);`
  - `echange(&a, &b);`
  - `echange(*a, *b);`

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101111
  - b. 111011010100110
  - c. 111011010100000
  - d. 111011010101000
17. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
18. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand *a* est un entier impair
  - c. N'affiche rien (quelque soit la valeur de *a*)
  - d. Affiche bonjour quand *a* est un entier pair
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
20. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

# Sujet n° 152

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. a pour valeur FAUX
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. l'adresse de la variable `a`
  - c. ce vers quoi pointe le pointeur `a`
  - d. n'a pas de sens
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 54
  - c. 68
  - d. 62
4. `printf("%i", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`
5. L'adresse d'une variable c'est :
  - a. ce vers quoi pointe la variable
  - b. l'adresse d'un pointeur sur la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. le contenu de la variable
6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(*a, *b) ;`
  - b. `echange(a++, b++) ;`
  - c. `echange(a, b) ;`
  - d. `echange(&a, &b) ;`
7. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Affiche bonjour quand `a` est un entier pair

8. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le code ASCII du caractère stocké dans la variable A
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
10. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - l'adresse de la variable `a`
11. Les instructions `i=0 ;`
- ```

while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
 - vont afficher 9 nombres
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
13. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un opérateur du langage C
 - Un mot-clef du langage C
14. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de `a`)
15. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
 - `c` vaut 'A'
 - `c` vaut 'a'
 - `c` vaut 'b'

16. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de a quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
17. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
18. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
20. L'expression `a<=1`
- a. Diminue de 1 la valeur de a
 - b. Réalise une affectation
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Utilise les opérateurs `<` et `=`

Sujet n° 153

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand a est un entier pair
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Affiche bonjour quand a est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
2. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
 - a. vont boucler indéfiniment
 - b. vont afficher 9 nombres
 - c. vont afficher 10 nombres
 - d. vont afficher 11 nombres
3. `if` est :
 - a. Un identificateur du langage C
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010101111
5. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 100
 - d. Permettent de d'affecter à `t[10]` la valeur 45
6. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A

7. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
8. L'expression `a<=1`
 - a. Diminue de 1 la valeur de `a`
 - b. Utilise les opérateurs `<` et `=`
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Réalise une affectation
9. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
10. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`
11. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
12. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur FAUX
 - c. n'a pas de valeur
 - d. a pour valeur VRAI
13. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - b. Affiche bonjour quelque soit `a`
 - c. Augmente la valeur de `a` quelque soit `a`
 - d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
14. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`

15. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

16. Après `char c; c='a'; c=c+1;`

- a. Un message d'erreur s'affiche
- b. `c` vaut `'A'`
- c. `c` vaut `'b'`
- d. `c` vaut `'a'`

17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. la valeur de `a` tout simplement
- c. n'a pas de sens
- d. l'adresse de la variable `a`

18. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand `a` est un entier pair
- b. N'affiche rien (quelque soit la valeur de `a`)
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand `a` est un entier impair

19. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 68
- b. 62
- c. 54
- d. 58

20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++);`
- b. `echange(a, b);`
- c. `echange(*a, *b);`
- d. `echange(&a, &b);`

Sujet n° 154

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
 - a. `echange(a++, b++) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(a, b) ;`
 - d. `echange(*a, *b) ;`
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
4. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
5. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Diminue de 1 la valeur de *a*
 - d. Réalise une affectation
6. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 10
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100

7. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
a. vont afficher 9 nombres
b. vont afficher 10 nombres
c. vont afficher 11 nombres
d. vont boucler indéfiniment
8. `if (a%2 == 0) printf("bonjour") ;`
a. Affiche bonjour quand a est un entier pair
b. Déclenche le message d'erreur `invalid lvalue in assignment`
c. N'affiche rien (quelque soit la valeur de a)
d. Affiche bonjour quand a est un entier impair
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
a. l'adresse de la variable `a`
b. ce vers quoi pointe le pointeur `a`
c. la valeur de `a` tout simplement
d. n'a pas de sens
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
a. 68
b. 58
c. 54
d. 62
12. `if (a<5) printf("Bonjour") ; a=a+1 ;`
a. Affiche bonjour et augmente la valeur de a quelque soit a
b. Augmente la valeur de a quelque soit a
c. Affiche bonjour quelque soit a
d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
13. `printf("%c", A)`
a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
b. Affiche le caractère `'A'`
c. Affiche le code ASCII du caractère stocké dans la variable `A`
d. Affiche le code ASCII du caractère `'A'`
14. `printf("%i", A)`
a. Affiche le code ASCII du caractère stocké dans la variable `A`
b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
c. Affiche le code ASCII du caractère `'A'`
d. Affiche le caractère `'A'`

15. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. l'adresse de la variable `a`
 - d. ce vers quoi pointe le pointeur `a`
17. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur VRAI
 - d. n'a pas de valeur
18. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Affiche bonjour quand `a` est un entier pair
 - d. Affiche bonjour quand `a` est un entier impair
19. `if` est :
- a. Un identificateur du langage C
 - b. Un mot-clef du langage C
 - c. Un opérateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
20. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
 - b. `c` vaut `'b'`
 - c. `c` vaut `'A'`
 - d. `c` vaut `'a'`

Sujet n° 155

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. Réalise une affectation
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Utilise les opérateurs `<` et `=`
 - d. Diminue de 1 la valeur de a
2. Les instructions

```
i=0 ;  
while(i<10)  
    printf("%i ", i) ;  
    i++ ;
```

 - a. vont afficher 9 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 10 nombres
 - d. vont afficher 11 nombres
3. Après `char c ; c='a' ; c=c+1 ;`
 - a. c vaut `'a'`
 - b. c vaut `'b'`
 - c. Un message d'erreur s'affiche
 - d. c vaut `'A'`
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010100110
5. `if` est :
 - a. Un opérateur du langage C
 - b. Un mot-clef du langage C
 - c. Un identificateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
6. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. a pour valeur VRAI
 - c. entraîne l'affichage d'un message d'erreur
 - d. n'a pas de valeur

7. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
8. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier pair
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 62
 - 68
 - 58
10. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - Affiche bonjour quelque soit *a*
11. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a, b);`
 - `echange(*a, *b);`
 - `echange(a++, b++);`
 - `echange(&a, &b);`
13. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
14. L'adresse d'une variable c'est :
- le contenu de la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable

15. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
16. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - ce vers quoi pointe le pointeur *a*
 - l'adresse de la variable *a*
 - la valeur de *a* tout simplement
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
18. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier pair
 - Affiche bonjour quand *a* est un entier impair
 - N'affiche rien (quelque soit la valeur de *a*)
19. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable *a*
 - n'a pas de sens
 - ce vers quoi pointe le pointeur *a*
 - la valeur de *a* tout simplement
20. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*

Sujet n° 156

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. Diminue de 1 la valeur de a
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Utilise les opérateurs `<` et `=`
2. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. a pour valeur VRAI
 - c. n'a pas de valeur
 - d. entraîne l'affichage d'un message d'erreur
3. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère `'A'`
4. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. la valeur de `a` tout simplement
5. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. le contenu de la variable
 - d. l'adresse d'un pointeur sur la variable
6. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. l'adresse de la variable `a`
 - d. ce vers quoi pointe le pointeur `a`

8. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
9. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
10. `printf("%c", A)`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
12. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'A'`
 - `c` vaut `'a'`
 - Un message d'erreur s'affiche
 - `c` vaut `'b'`
13. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand `a` est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
 - N'affiche rien (quelque soit la valeur de `a`)
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 62
 - 68
 - 58
15. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier pair

16. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- a. vont afficher 10 nombres
 - b. vont afficher 11 nombres
 - c. vont boucler indéfiniment
 - d. vont afficher 9 nombres
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
 - b. 111011010101111
 - c. 111011010101000
 - d. 111011010100000
18. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
19. `if` est :
- a. Un identificateur du langage C
 - b. Un mot-clef du langage C
 - c. Un opérateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
20. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - b. Affiche bonjour quelque soit *a*
 - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - d. Augmente la valeur de *a* quelque soit *a*

Sujet n° 157

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'A'`
 - b. `c` vaut `'b'`
 - c. `c` vaut `'a'`
 - d. Un message d'erreur s'affiche
2. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Augmente la valeur de `a` quelque soit `a`
 - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - c. Affiche bonjour quelque soit `a`
 - d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. n'a pas de valeur
 - d. a pour valeur FAUX
4. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le numéro de la case mémoire où le contenu de la variable est stocké
 - c. ce vers quoi pointe la variable
 - d. le contenu de la variable
5. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. ce vers quoi pointe le pointeur `a`
 - b. n'a pas de sens
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
7. L'expression `a<=1`
 - a. Diminue de 1 la valeur de `a`
 - b. Utilise les opérateurs `<` et `=`
 - c. Réalise une affectation
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon

8. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
9. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier pair
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a, b);`
 - `echange(a++, b++);`
 - `echange(*a, *b);`
 - `echange(&a, &b);`
11. Les instructions `i=0;`
- ```
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010100110
  - 111011010101111
  - 111011010101000
13. `if` est :
- Un identificateur du langage C
  - Un opérateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
14. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
15. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable *a*
  - la valeur de *a* tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur *a*



16. `printf("%c", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
17. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 45
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 100
18. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand *a* est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de *a*)
  - d. Affiche bonjour quand *a* est un entier pair
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 68
  - b. 62
  - c. 54
  - d. 58

# Sujet n° 158

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand  $a$  est un entier impair
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
2. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Augmente la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour quelque soit  $a$
  - d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. Affiche bonjour quand  $a$  est un entier pair
4. Après `char c ; c='a' ; c=c+1 ;`
  - a.  $c$  vaut `'a'`
  - b.  $c$  vaut `'A'`
  - c. Un message d'erreur s'affiche
  - d.  $c$  vaut `'b'`
5. On suppose que  $a$  a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. ce vers quoi pointe le pointeur  $a$
  - b. l'adresse de la variable  $a$
  - c. la valeur de  $a$  tout simplement
  - d. n'a pas de sens
6. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 62
  - c. 54
  - d. 58
7. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$

8. L'expression `a<=1`
- Réalise une affectation
  - Diminue de 1 la valeur de  $a$
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur
  - n'a pas de valeur
  - a pour valeur VRAI
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
  - 111011010101111
  - 111011010100000
  - 111011010100110
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
13. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le contenu de la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
14. Les instructions `i=0;`
- ```

while(i<10)
    printf("%i ", i);
    i++;

```
- vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens

16. `printf("%c", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(&a, &b) ;`
 - `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
 - `echange(a, b) ;`
18. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
19. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
20. `if` est :
- Un mot-clef du langage C
 - Un opérateur du langage C
 - Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande

Sujet n° 159

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101111
- c. 111011010101000
- d. 111011010100110

2. `if (a%2 == 0) printf("bonjour");`

- a. N'affiche rien (quelque soit la valeur de *a*)
- b. Affiche bonjour quand *a* est un entier pair
- c. Affiche bonjour quand *a* est un entier impair
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

3. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

4. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le code ASCII du caractère stocké dans la variable *A*

5. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 11 nombres
- c. vont afficher 9 nombres
- d. vont afficher 10 nombres

6. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :

- a. `echange(*a, *b);`
- b. `echange(a, b);`
- c. `echange(&a, &b);`
- d. `echange(a++, b++);`

7. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier impair
 - Affiche bonjour quand a est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
8. `printf("%i", 'A')`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère stocké dans la variable A
9. L'expression `a--`
- Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de a
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
10. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un identificateur du langage C
 - Un opérateur du langage C
11. Après `char c; c='a'; c=c+1;`
- c vaut 'b'
 - c vaut 'a'
 - c vaut 'A'
 - Un message d'erreur s'affiche
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
13. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de a quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
 - Augmente la valeur de a quelque soit a
14. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
 - ce vers quoi pointe la variable
15. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement

16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur VRAI
 - c. a pour valeur FAUX
 - d. n'a pas de valeur
17. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 0
 - c. Permettent de d'affecter à `t[10]` la valeur 100
 - d. Permettent de d'affecter à `t[10]` la valeur 10
20. `printf("%c", A)`
- a. Affiche le code ASCII du caractère '`A`'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère '`A`'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

Sujet n° 160

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère 'A'
2. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 58
 - c. 54
 - d. 62
3. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
4. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un opérateur du langage C
 - c. Un identificateur du langage C
 - d. Un mot-clef du langage C
5. L'expression `a<=1`
 - a. Utilise les opérateurs `<` et `=`
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de *a*
6. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 0
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 10
7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`

8. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable
9. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'b'`
 - `c` vaut `'A'`
 - Un message d'erreur s'affiche
 - `c` vaut `'a'`
10. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
13. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur VRAI
  - n'a pas de valeur
15. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
  - N'affiche rien (quelque soit la valeur de `a`)
  - Affiche bonjour quand `a` est un entier impair

16. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère 'A'
17. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - b. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - c. Affiche bonjour quelque soit *a*
  - d. Augmente la valeur de *a* quelque soit *a*
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
  - b. 111011010101000
  - c. 111011010101111
  - d. 111011010100000
19. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand *a* est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand *a* est un entier pair
  - d. N'affiche rien (quelque soit la valeur de *a*)
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- a. `echange(a++, b++)` ;
  - b. `echange(&a, &b)` ;
  - c. `echange(*a, *b)` ;
  - d. `echange(a, b)` ;



# Sujet n° 161

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'b'`
  - b. `c` vaut `'a'`
  - c. `c` vaut `'A'`
  - d. Un message d'erreur s'affiche
2. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. n'a pas de valeur
  - b. a pour valeur FAUX
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur VRAI
3. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de `a`)
  - c. Affiche bonjour quand `a` est un entier impair
  - d. Affiche bonjour quand `a` est un entier pair
4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(&a, &b) ;`
  - b. `echange(a++, b++) ;`
  - c. `echange(a, b) ;`
  - d. `echange(*a, *b) ;`
5. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
6. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
7. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier impair
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier pair

8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 68
  - 54
  - 58
9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
  - 111011010100110
  - 111011010101000
  - 111011010100000
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
14. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
15. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère '`A`'

16. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
17. L'expression `a<=1`
- Diminue de 1 la valeur de *a*
  - Réalise une affectation
  - Utilise les opérateurs < et =
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
18. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont afficher 10 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 11 nombres
19. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère 'A'
20. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
  - Un opérateur du langage C
  - Un mot-clef du langage C

# Sujet n° 162

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010100000
- c. 111011010101111
- d. 111011010101000

2. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit  $a$
- b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
- c. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
- d. Augmente la valeur de  $a$  quelque soit  $a$

3. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable  $A$
- b. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le caractère 'A'

4. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- d. Affiche le code ASCII du caractère stocké dans la variable  $A$

5. On suppose que  $a$  a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur  $a$
- b. la valeur de  $a$  tout simplement
- c. l'adresse de la variable  $a$
- d. n'a pas de sens

6. L'adresse d'une variable c'est :

- a. ce vers quoi pointe la variable
- b. le contenu de la variable
- c. le numéro de la case mémoire où le contenu de la variable est stocké
- d. l'adresse d'un pointeur sur la variable

7. `printf("%i", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- b. Affiche le code ASCII du caractère stocké dans la variable  $A$
- c. Affiche le code ASCII du caractère 'A'
- d. Affiche le caractère 'A'



8. L'expression `a<=1`
  - a. Réalise une affectation
  - b. Diminue de 1 la valeur de  $a$
  - c. Utilise les opérateurs `<` et `=`
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 54
  - c. 58
  - d. 68
10. `if (a%2 == 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
11. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
12. `if` est :
  - a. Un opérateur du langage C
  - b. Un identificateur du langage C
  - c. Un mot-clef du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
13. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`
14. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a, b);`
  - b. `echange(a++, b++);`
  - c. `echange(*a, *b);`
  - d. `echange(&a, &b);`
15. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 0
  - c. Permettent de d'affecter à `t[10]` la valeur 10
  - d. Permettent de d'affecter à `t[10]` la valeur 45

16. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Affiche bonjour quand `a` est un entier impair
  - d. N'affiche rien (quelque soit la valeur de `a`)
17. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
  - b. `c` vaut `'b'`
  - c. `c` vaut `'a'`
  - d. `c` vaut `'A'`
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur VRAI
  - c. a pour valeur FAUX
  - d. n'a pas de valeur
20. Les instructions
- ```
i=0;
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont afficher 10 nombres
 - b. vont afficher 11 nombres
 - c. vont afficher 9 nombres
 - d. vont boucler indéfiniment

Sujet n° 163

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
 - a. Un identificateur du langage C
 - b. Un opérateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
2. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Affiche bonjour quelque soit a
3. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère 'A'
4. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 58
 - c. 62
 - d. 54
6. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère 'A'
7. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère 'A'

8. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand a est un entier pair
 - Affiche bonjour quand a est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
10. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
11. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
 - c vaut 'a'
 - c vaut 'A'
 - c vaut 'b'
12. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de a
13. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - le contenu de la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - n'a pas de valeur
15. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
 - 111011010101111
 - 111011010101000
 - 111011010100000

16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(&a, &b) ;`
- c. `echange(a, b) ;`
- d. `echange(*a, *b) ;`

17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. l'adresse de la variable `a`
- c. n'a pas de sens
- d. la valeur de `a` tout simplement

18. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 100
- d. Permettent de d'affecter à `t[10]` la valeur 0

19. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier pair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. Affiche bonjour quand `a` est un entier impair
- d. N'affiche rien (quelque soit la valeur de `a`)

20. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 9 nombres

Sujet n° 164

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Réalise une affectation
 - c. Diminue de 1 la valeur de a
 - d. Utilise les opérateurs `<` et `=`
2. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier pair
 - c. N'affiche rien (quelque soit la valeur de a)
 - d. Affiche bonjour quand a est un entier impair
3. `printf("%c", 'A')`
 - a. Affiche le caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable A
4. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
5. `if` est :
 - a. Un opérateur du langage C
 - b. Un identificateur du langage C
 - c. Un mot-clef du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
6. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 68
 - b. 54
 - c. 58
 - d. 62
7. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère `'A'`

8. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
9. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère stocké dans la variable A
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
12. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur VRAI
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur FAUX
 - d. n'a pas de valeur
13. Après `char c; c='a'; c=c+1;`
 - a. c vaut 'A'
 - b. Un message d'erreur s'affiche
 - c. c vaut 'b'
 - d. c vaut 'a'
14. `if (a%2 == 0) printf("bonjour");`
 - a. N'affiche rien (quelque soit la valeur de a)
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier pair
 - d. Affiche bonjour quand a est un entier impair
15. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Augmente la valeur de a quelque soit a

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
17. `printf("%c", A)`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
18. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i) ;
 i++ ;
```
- vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 11 nombres
  - vont afficher 10 nombres
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b) ;`
  - `echange(*a, *b) ;`
  - `echange(a++, b++) ;`
  - `echange(a, b) ;`
20. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0

# Sujet n° 165

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `a<=1`
  - a. Diminue de 1 la valeur de  $a$
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Réalise une affectation
2. `if (a%2 == 0) printf("bonjour");`
  - a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. Affiche bonjour quand  $a$  est un entier pair
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 62
  - b. 58
  - c. 68
  - d. 54
4. `if` est :
  - a. Un mot-clef du langage C
  - b. Une commande qu'on tape dans la fenêtre de commande
  - c. Un identificateur du langage C
  - d. Un opérateur du langage C
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. la valeur de `a` tout simplement
  - c. ce vers quoi pointe le pointeur `a`
  - d. n'a pas de sens
6. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Augmente la valeur de  $a$  quelque soit  $a$
  - c. Affiche bonjour quelque soit  $a$
  - d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
7. `printf("%i", 'A')`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère `'A'`
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - d. Affiche le code ASCII du caractère stocké dans la variable `A`

8. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
10. Les instructions `i=0;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
 - vont afficher 11 nombres
11. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
12. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
13. `printf("%c", A)`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
14. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur FAUX
 - a pour valeur VRAI

16. L'adresse d'une variable c'est :
- a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. ce vers quoi pointe la variable
17. Après `char c ; c='a' ; c=c+1 ;`
- a. Un message d'erreur s'affiche
 - b. `c` vaut `'A'`
 - c. `c` vaut `'a'`
 - d. `c` vaut `'b'`
18. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b) ;`
 - b. `echange(&a, &b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(a++, b++) ;`
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
 - b. 111011010100110
 - c. 111011010101111
 - d. 111011010100000
20. `printf("%i", A)`
- a. Affiche le caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`

Sujet n° 166

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 11 nombres
 - b. vont afficher 10 nombres
 - c. vont afficher 9 nombres
 - d. vont boucler indéfiniment
2. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand a est un entier pair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier impair
 - d. N'affiche rien (quelque soit la valeur de a)
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010100110
4. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 45
 - d. Permettent de d'affecter à `t[10]` la valeur 0
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(a++, b++) ;`
 - b. `echange(a, b) ;`
 - c. `echange(*a, *b) ;`
 - d. `echange(&a, &b) ;`
6. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
 - c. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
 - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`

7. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
8. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand *a* est un entier pair
 - b. Affiche bonjour quand *a* est un entier impair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
9. Après `char c; c='a'; c=c+1;`
 - a. *c* vaut 'b'
 - b. *c* vaut 'a'
 - c. Un message d'erreur s'affiche
 - d. *c* vaut 'A'
10. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le caractère 'A'
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 54
 - b. 58
 - c. 68
 - d. 62
12. L'expression `a<=1`
 - a. Diminue de 1 la valeur de *a*
 - b. Réalise une affectation
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Utilise les opérateurs `<` et `=`
13. L'adresse d'une variable c'est :
 - a. ce vers quoi pointe la variable
 - b. le contenu de la variable
 - c. l'adresse d'un pointeur sur la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
14. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit *a*
 - b. Augmente la valeur de *a* quelque soit *a*
 - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
15. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. n'a pas de valeur
 - c. a pour valeur VRAI
 - d. a pour valeur FAUX

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens
17. `if` est :
- a. Un mot-clef du langage C
 - b. Un identificateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un opérateur du langage C
18. `printf("%c", 'A')`
- a. Affiche le caractère `'A'`
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
20. `printf("%c", A)`
- a. Affiche le caractère `'A'`
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère `'A'`

Sujet n° 167

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
2. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand *a* est un entier impair
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
3. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable *a*
 - b. la valeur de *a* tout simplement
 - c. ce vers quoi pointe le pointeur *a*
 - d. n'a pas de sens
4. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit *a*
 - b. Augmente la valeur de *a* quelque soit *a*
 - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
5. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère stocké dans la variable *A*
 - d. Affiche le caractère 'A'
6. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010101111
 - b. 111011010100000
 - c. 111011010100110
 - d. 111011010101000
7. Après `char c; c='a'; c=c+1;`
 - a. Un message d'erreur s'affiche
 - b. *c* vaut 'a'
 - c. *c* vaut 'A'
 - d. *c* vaut 'b'

8. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
9. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 68
 - 62
 - 54
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur VRAI
 - a pour valeur FAUX
12. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
13. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
 - la valeur de *a* tout simplement
 - l'adresse de la variable *a*
 - n'a pas de sens
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
15. `if` est :
- Un opérateur du langage C
 - Un mot-clef du langage C
 - Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande

16. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
 - vont afficher 10 nombres
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(a, b) ;`
 - `echange(&a, &b) ;`
18. L'expression `a<=1`
- Réalise une affectation
 - Diminue de 1 la valeur de `a`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
19. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 45
20. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable

Sujet n° 168

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'b'`
 - b. `c` vaut `'a'`
 - c. `c` vaut `'A'`
 - d. Un message d'erreur s'affiche
2. `if (a%2 == 0) printf("bonjour") ;`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand `a` est un entier pair
 - c. Affiche bonjour quand `a` est un entier impair
 - d. N'affiche rien (quelque soit la valeur de `a`)
3. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand `a` est un entier impair
 - b. Affiche bonjour quand `a` est un entier pair
 - c. N'affiche rien (quelque soit la valeur de `a`)
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
4. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens
5. L'expression `a--`
 - a. Réalise une affectation
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Utilise les opérateurs `<` et `=`
 - d. Diminue de 1 la valeur de `a`
6. `printf("%c", 'A')`
 - a. Affiche le caractère `'A'`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
7. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. n'a pas de sens
 - b. ce vers quoi pointe le pointeur `a`
 - c. la valeur de `a` tout simplement
 - d. l'adresse de la variable `a`

8. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
9. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
10. L'adresse d'une variable c'est :
- ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable
11. `if` est :
- Un opérateur du langage C
 - Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
13. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 62
 - 58
 - 68
14. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
 - a pour valeur VRAI
 - a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur

16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b) ;`
- b. `echange(a++, b++) ;`
- c. `echange(&a, &b) ;`
- d. `echange(a, b) ;`

17. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

18. `printf("%c", A)`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le code ASCII du caractère `'A'`

19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101000
- c. 111011010100000
- d. 111011010101111

20. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i) ;
    i++ ;
```

- a. vont afficher 9 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 11 nombres

Sujet n° 169

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

2. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. ce vers quoi pointe le pointeur `a`
- d. l'adresse de la variable `a`

3. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable `A`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

4. `if (a<5) printf("Bonjour"); a=a+1;`

- a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- c. Affiche bonjour quelque soit `a`
- d. Augmente la valeur de `a` quelque soit `a`

5. `printf("%i", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

6. L'adresse d'une variable c'est :

- a. le contenu de la variable
- b. le numéro de la case mémoire où le contenu de la variable est stocké
- c. l'adresse d'un pointeur sur la variable
- d. ce vers quoi pointe la variable

7. L'expression `a<=1`

- a. Diminue de 1 la valeur de `a`
- b. Utilise les opérateurs `<` et `=`
- c. Réalise une affectation
- d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon

8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
 - 68
 - 62
 - 58
9. `if` est :
- Un opérateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un mot-clef du langage C
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de `a`)
11. `printf("%c", 'A')`
- Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0
13. `printf("%c", A)`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- `a` pour valeur FAUX
 - n'a pas de valeur
 - entraîne l'affichage d'un message d'erreur
 - `a` pour valeur VRAI
15. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres
 - vont afficher 9 nombres

16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100110
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010101000
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(&a, &b) ;`
 - b. `echange(a, b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(*a, *b) ;`
18. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand `a` est un entier pair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand `a` est un entier impair
 - d. N'affiche rien (quelque soit la valeur de `a`)
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
 - b. l'adresse de la variable `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens
20. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'a'`
 - b. Un message d'erreur s'affiche
 - c. `c` vaut `'A'`
 - d. `c` vaut `'b'`

Sujet n° 170

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 100
2. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
3. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. a pour valeur FAUX
 - b. entraîne l'affichage d'un message d'erreur
 - c. n'a pas de valeur
 - d. a pour valeur VRAI
4. Après `char c ; c='a' ; c=c+1 ;`
 - a. Un message d'erreur s'affiche
 - b. `c` vaut 'A'
 - c. `c` vaut 'b'
 - d. `c` vaut 'a'
5. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le caractère 'A'
6. `printf("%i", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable `A`
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
7. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 58
 - c. 54
 - d. 68

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
9. `if` est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
 - Un identificateur du langage C
 - Un mot-clef du langage C
10. L'expression `a<=1`
- Diminue de 1 la valeur de `a`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
 - Réalise une affectation
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
12. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
13. Les instructions
- ```
i=0 ;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont boucler indéfiniment
  - vont afficher 9 nombres
  - vont afficher 10 nombres
  - vont afficher 11 nombres
14. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101111
  - 111011010100110
  - 111011010101000

15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(*a, *b) ;`
- c. `echange(&a, &b) ;`
- d. `echange(a, b) ;`

16. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. l'adresse de la variable `a`
- c. n'a pas de sens
- d. la valeur de `a` tout simplement

17. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier impair
- b. Affiche bonjour quand `a` est un entier pair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. N'affiche rien (quelque soit la valeur de `a`)

18. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère `'A'`

19. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Augmente la valeur de `a` quelque soit `a`
- b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- c. Affiche bonjour quelque soit `a`
- d. Affiche bonjour et augmente la valeur de `a` quelque soit `a`

20. `if (a%2 == 0) printf("bonjour") ;`

- a. N'affiche rien (quelque soit la valeur de `a`)
- b. Affiche bonjour quand `a` est un entier impair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand `a` est un entier pair

# Sujet n° 171

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un opérateur du langage C
  - c. Un identificateur du langage C
  - d. Un mot-clef du langage C
2. Après `char c ; c='a' ; c=c+1 ;`
  - a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'A'`
  - d. `c` vaut `'b'`
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. n'a pas de sens
  - b. la valeur de `a` tout simplement
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
4. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier impair
5. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. n'a pas de valeur
  - b. a pour valeur VRAI
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur FAUX
6. `printf("%i", A)`
  - a. Affiche le caractère `'A'`
  - b. Affiche le code ASCII du caractère `'A'`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
7. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. ce vers quoi pointe la variable
  - d. le contenu de la variable

8. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(&a, &b) ;`
- d. `echange(*a, *b) ;`

9. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère `'A'`

10. `if (a%2 == 0) printf("bonjour") ;`

- a. Affiche bonjour quand `a` est un entier pair
- b. Déclenche le message d'erreur `invalid lvalue in assignment`
- c. N'affiche rien (quelque soit la valeur de `a`)
- d. Affiche bonjour quand `a` est un entier impair

11. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

12. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Diminue de 1 la valeur de `a`
- c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- d. Réalise une affectation

13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

14. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101000
- c. 111011010101111
- d. 111011010100110

15. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère `'A'`

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - n'a pas de sens
17. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i);
    i++ ;
```
- vont afficher 9 nombres
 - vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
18. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- Affiche bonjour quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - Augmente la valeur de *a* quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
19. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
20. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100

Sujet n° 172

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. la valeur de `a` tout simplement
2. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - b. Affiche le caractère `'A'`
 - c. Affiche le code ASCII du caractère `'A'`
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
3. `printf("%i", A)`
 - a. Affiche le caractère `'A'`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
4. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
 - a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 10
5. L'expression `a<=1`
 - a. Diminue de 1 la valeur de `a`
 - b. Réalise une affectation
 - c. Utilise les opérateurs `<` et `=`
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
6. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. l'adresse d'un pointeur sur la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
7. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'a'`
 - b. `c` vaut `'A'`
 - c. `c` vaut `'b'`
 - d. Un message d'erreur s'affiche

8. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - n'a pas de sens
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
10. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b);`
 - `echange(a++, b++);`
 - `echange(&a, &b);`
 - `echange(*a, *b);`
12. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 68
 - 54
 - 62
13. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Augmente la valeur de `a` quelque soit `a`
14. `printf("%c", 'A')`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
15. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier impair
 - Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

17. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 11 nombres
- c. vont afficher 10 nombres
- d. vont afficher 9 nombres

18. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur VRAI
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur FAUX

19. `if` est :

- a. Un identificateur du langage C
- b. Un mot-clef du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un opérateur du langage C

20. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101111
- b. 111011010100000
- c. 111011010101000
- d. 111011010100110

Sujet n° 173

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - b. Affiche bonjour quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. Augmente la valeur de a quelque soit a
2. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère 'A'
3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010101000
 - c. 111011010100000
 - d. 111011010101111
4. `printf("%i", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
5. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
 - a. Permettent de d'affecter à $t[10]$ la valeur 10
 - b. Permettent de d'affecter à $t[10]$ la valeur 45
 - c. Permettent de d'affecter à $t[10]$ la valeur 100
 - d. Permettent de d'affecter à $t[10]$ la valeur 0
6. Après `char c; c='a'; c=c+1;`
 - a. c vaut 'b'
 - b. c vaut 'A'
 - c. c vaut 'a'
 - d. Un message d'erreur s'affiche
7. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le caractère 'A'

8. L'expression `a<=1`
- Réalise une affectation
 - Diminue de 1 la valeur de `a`
 - Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
9. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
11. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
12. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - le contenu de la variable
 - ce vers quoi pointe la variable
13. `if` est :
- Un mot-clef du langage C
 - Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 68
 - 58
 - 54
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - n'a pas de valeur

16. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Affiche bonjour quand `a` est un entier impair
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - n'a pas de sens
18. `if (a%2 = 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de `a`)
19. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
20. Les instructions
- ```
i=0;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 10 nombres
  - vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment

# Sujet n° 174

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour"); a=a+1;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - c. Augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour quelque soit  $a$
2. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
3. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
4. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 0
5. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable  $A$
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - d. Affiche le caractère 'A'
6. `if (a%2 == 0) printf("bonjour");`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Affiche bonjour quand  $a$  est un entier impair
  - d. Affiche bonjour quand  $a$  est un entier pair

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(&a, &b) ;`
- b. `echange(a, b) ;`
- c. `echange(a++, b++) ;`
- d. `echange(*a, *b) ;`

8. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. le numéro de la case mémoire où le contenu de la variable est stocké
- c. le contenu de la variable
- d. ce vers quoi pointe la variable

9. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Diminue de 1 la valeur de  $a$
- d. Réalise une affectation

10. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le code ASCII du caractère stocké dans la variable  $A$
- d. Affiche le caractère `'A'`

11. `if` est :

- a. Un identificateur du langage C
- b. Un opérateur du langage C
- c. Un mot-clef du langage C
- d. Une commande qu'on tape dans la fenêtre de commande

12. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable  $A$
- c. Affiche le caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$

13. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. entraîne l'affichage d'un message d'erreur
- b. a pour valeur VRAI
- c. a pour valeur FAUX
- d. n'a pas de valeur

14. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```

- a. vont afficher 9 nombres
- b. vont boucler indéfiniment
- c. vont afficher 10 nombres
- d. vont afficher 11 nombres

15. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de *a*)
  - d. Affiche bonjour quand a est un entier pair
16. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101111
  - b. 111011010100000
  - c. 111011010101000
  - d. 111011010100110
17. On suppose que **a** a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
  - b. la valeur de **a** tout simplement
  - c. l'adresse de la variable **a**
  - d. ce vers quoi pointe le pointeur **a**
18. On suppose que **a** a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable **a**
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur **a**
  - d. la valeur de **a** tout simplement
19. Après `char c; c='a'; c=c+1;`
- a. *c* vaut 'b'
  - b. *c* vaut 'a'
  - c. Un message d'erreur s'affiche
  - d. *c* vaut 'A'
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 54
  - b. 68
  - c. 62
  - d. 58

# Sujet n° 175

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 10
  - b. Permettent de d'affecter à `t[10]` la valeur 45
  - c. Permettent de d'affecter à `t[10]` la valeur 0
  - d. Permettent de d'affecter à `t[10]` la valeur 100
2. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t ; t=*a ; *a=*b ; *b=t ;}`
  - b. `void echange(int &a, int &b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - c. `void echange(int *a, int *b) {int t ; t=&a ; &a=&b ; &b=t ;}`
  - d. `void echange(int &a, int &b) {int t ; t=*a ; *a=*b ; *b=t ;}`
3. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur VRAI
  - b. entraîne l'affichage d'un message d'erreur
  - c. a pour valeur FAUX
  - d. n'a pas de valeur
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100000
  - b. 111011010100110
  - c. 111011010101000
  - d. 111011010101111
5. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 54
  - c. 62
  - d. 58
6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. ce vers quoi pointe le pointeur `a`
  - b. la valeur de `a` tout simplement
  - c. n'a pas de sens
  - d. l'adresse de la variable `a`

7. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- vont afficher 11 nombres
  - vont afficher 10 nombres
  - vont boucler indéfiniment
  - vont afficher 9 nombres
8. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable A
  - Affiche le code ASCII du caractère stocké dans la variable A
9. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - le contenu de la variable
10. L'expression `a<=1`
- Diminue de 1 la valeur de  $a$
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
11. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
  - N'affiche rien (quelque soit la valeur de  $a$ )
12. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - $c$  vaut 'b'
  - $c$  vaut 'a'
  - $c$  vaut 'A'
13. `if (a%2 = 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
14. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$

15. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
16. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le caractère `'A'`
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens
18. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b) ;`
  - `echange(&a, &b) ;`
  - `echange(a++, b++) ;`
  - `echange(*a, *b) ;`
19. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
20. `if` est :
- Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un mot-clef du langage C
  - Un opérateur du langage C

# Sujet n° 176

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. l'adresse de la variable `a`
- b. la valeur de `a` tout simplement
- c. n'a pas de sens
- d. ce vers quoi pointe le pointeur `a`

2. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i);
 i++;
```

- a. vont afficher 11 nombres
- b. vont boucler indéfiniment
- c. vont afficher 9 nombres
- d. vont afficher 10 nombres

3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. ce vers quoi pointe le pointeur `a`
- c. la valeur de `a` tout simplement
- d. l'adresse de la variable `a`

4. L'adresse d'une variable c'est :

- a. l'adresse d'un pointeur sur la variable
- b. le contenu de la variable
- c. le numéro de la case mémoire où le contenu de la variable est stocké
- d. ce vers quoi pointe la variable

5. `printf("%i", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

6. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- b. Augmente la valeur de `a` quelque soit `a`
- c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- d. Affiche bonjour quelque soit `a`

7. `if` est :
- Un opérateur du langage C
  - Un mot-clef du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
8. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de  $a$ )
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier pair
  - Affiche bonjour quand  $a$  est un entier impair
9. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le caractère 'A'
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 54
  - 68
  - 62
11. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
- `echange(*a, *b) ;`
  - `echange(a, b) ;`
  - `echange(&a, &b) ;`
  - `echange(a++, b++) ;`
12. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
13. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
14. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
  - entraîne l'affichage d'un message d'erreur
  - n'a pas de valeur
  - a pour valeur VRAI

15. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier pair
  - N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier impair
16. Après `char c; c='a'; c=c+1;`
- $c$  vaut `'b'`
  - Un message d'erreur s'affiche
  - $c$  vaut `'A'`
  - $c$  vaut `'a'`
17. L'expression `a<=1`
- A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Réalise une affectation
  - Diminue de 1 la valeur de  $a$
  - Utilise les opérateurs `<` et `=`
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
  - 111011010100000
  - 111011010101111
  - 111011010100110
19. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`



# Sujet n° 177

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 0
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 100
2. `printf("%i", 'A')`
  - a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
3. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
4. L'expression `a<=1`
  - a. Diminue de 1 la valeur de *a*
  - b. Utilise les opérateurs `<` et `=`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Réalise une affectation
5. `if` est :
  - a. Une commande qu'on tape dans la fenêtre de commande
  - b. Un mot-clef du langage C
  - c. Un opérateur du langage C
  - d. Un identificateur du langage C
6. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
7. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. a pour valeur VRAI
  - d. entraîne l'affichage d'un message d'erreur

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
  - l'adresse de la variable `a`
9. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
  - 54
  - 62
  - 58
10. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`
  - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
12. `printf("%i", A)`
- Affiche le caractère '`A`'
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère '`A`'
  - Affiche le code ASCII du caractère stocké dans la variable `A`
13. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
  - le numéro de la case mémoire où le contenu de la variable est stocké
  - ce vers quoi pointe la variable
  - le contenu de la variable
14. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
  - `c` vaut '`b`'
  - `c` vaut '`a`'
  - `c` vaut '`A`'
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(&a, &b);`
  - `echange(a, b);`
  - `echange(*a, *b);`
  - `echange(a++, b++);`

16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier pair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier impair
17. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable `a`
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. n'a pas de sens
18. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier impair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
19. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010100000
  - b. 111011010101000
  - c. 111011010100110
  - d. 111011010101111
20. Les instructions
- ```
i=0 ;
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont boucler indéfiniment
 - b. vont afficher 9 nombres
 - c. vont afficher 11 nombres
 - d. vont afficher 10 nombres

Sujet n° 178

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. n'a pas de valeur
 - b. a pour valeur FAUX
 - c. entraîne l'affichage d'un message d'erreur
 - d. a pour valeur VRAI
2. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 58
 - b. 54
 - c. 68
 - d. 62
3. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. ce vers quoi pointe la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. l'adresse d'un pointeur sur la variable
4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
5. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour et augmente la valeur de a quelque soit a
 - b. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - c. Affiche bonjour quelque soit a
 - d. Augmente la valeur de a quelque soit a
6. `printf("%c", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable A
7. Après `char c; c='a'; c=c+1;`
 - a. Un message d'erreur s'affiche
 - b. c vaut 'a'
 - c. c vaut 'b'
 - d. c vaut 'A'

8. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
 - l'adresse de la variable `a`
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010101000
 - 111011010101111
 - 111011010100110
10. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand `a` est un entier impair
 - Affiche bonjour quand `a` est un entier pair
 - N'affiche rien (quelque soit la valeur de `a`)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
11. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
12. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b);`
 - `echange(a, b);`
 - `echange(&a, &b);`
 - `echange(a++, b++);`
13. `printf("%i", A)`
- Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable `A`
14. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère 'A'
15. Les instructions
- ```
i=0;
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres

16. `if` est :
- a. Un opérateur du langage C
  - b. Un mot-clef du langage C
  - c. Un identificateur du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
17. L'expression `a<=1`
- a. Utilise les opérateurs `<` et `=`
  - b. Réalise une affectation
  - c. Diminue de 1 la valeur de  $a$
  - d. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
  - b. l'adresse de la variable `a`
  - c. la valeur de `a` tout simplement
  - d. ce vers quoi pointe le pointeur `a`
19. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
20. `printf("%i", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - b. Affiche le code ASCII du caractère stocké dans la variable `A`
  - c. Affiche le caractère `'A'`
  - d. Affiche le code ASCII du caractère `'A'`

# Sujet n° 179

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
2. L'expression `a<=1`
  - a. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - b. Réalise une affectation
  - c. Diminue de 1 la valeur de *a*
  - d. Utilise les opérateurs `<` et `=`
3. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. l'adresse d'un pointeur sur la variable
  - c. le contenu de la variable
  - d. ce vers quoi pointe la variable
4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
  - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
5. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. a pour valeur VRAI
  - c. n'a pas de valeur
  - d. entraîne l'affichage d'un message d'erreur
6. `if (a%2 == 0) printf("bonjour");`
  - a. Affiche bonjour quand *a* est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de *a*)
  - d. Affiche bonjour quand *a* est un entier pair
7. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 68
  - b. 54
  - c. 62
  - d. 58

8. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
9. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010100110
  - 111011010101000
  - 111011010101111
11. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
12. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- la valeur de *a* tout simplement
  - n'a pas de sens
  - l'adresse de la variable *a*
  - ce vers quoi pointe le pointeur *a*
13. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de *a* tout simplement
  - n'a pas de sens
  - l'adresse de la variable *a*
  - ce vers quoi pointe le pointeur *a*
14. Les instructions `i=0 ; while(i<10) printf("%i ", i) ; i++ ;`
- vont boucler indéfiniment
  - vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont afficher 10 nombres
15. `printf("%i", A)`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'

16. Après `char c ; c='a' ; c=c+1 ;`
- a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'A'`
  - d. `c` vaut `'b'`
17. `if (a%2 == 0) printf("bonjour") ;`
- a. Affiche bonjour quand `a` est un entier impair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier pair
18. `if` est :
- a. Un opérateur du langage C
  - b. Un mot-clef du langage C
  - c. Un identificateur du langage C
  - d. Une commande qu'on tape dans la fenêtre de commande
19. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Augmente la valeur de `a` quelque soit `a`
  - b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - c. Affiche bonjour quelque soit `a`
  - d. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b) ;`
  - b. `echange(a++, b++) ;`
  - c. `echange(&a, &b) ;`
  - d. `echange(*a, *b) ;`

# Sujet n° 180

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.



## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
  - a. `echange(*a, *b) ;`
  - b. `echange(a, b) ;`
  - c. `echange(a++, b++) ;`
  - d. `echange(&a, &b) ;`
3. Après `char c ; c='a' ; c=c+1 ;`
  - a. *c* vaut 'b'
  - b. *c* vaut 'a'
  - c. Un message d'erreur s'affiche
  - d. *c* vaut 'A'
4. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. a pour valeur VRAI
  - c. n'a pas de valeur
  - d. entraîne l'affichage d'un message d'erreur
5. `printf("%c", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
6. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
  - a. Permettent de d'affecter à `t[10]` la valeur 100
  - b. Permettent de d'affecter à `t[10]` la valeur 10
  - c. Permettent de d'affecter à `t[10]` la valeur 45
  - d. Permettent de d'affecter à `t[10]` la valeur 0
7. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - b. Augmente la valeur de *a* quelque soit *a*
  - c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - d. Affiche bonjour quelque soit *a*

8. `if` est :
- Un mot-clef du langage C
  - Un opérateur du langage C
  - Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101000
  - 111011010100110
  - 111011010101111
10. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
  - l'adresse de la variable `a`
  - n'a pas de sens
  - ce vers quoi pointe le pointeur `a`
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 58
  - 68
12. L'expression `a<=1`
- Réalise une affectation
  - Diminue de 1 la valeur de `a`
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
13. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
14. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de `a`)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier impair
  - Affiche bonjour quand `a` est un entier pair
15. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
  - ce vers quoi pointe le pointeur `a`
  - la valeur de `a` tout simplement
  - n'a pas de sens

16. Les instructions `i=0 ;`  
`while(i<10)`  
`printf("%i ", i) ;`  
`i++ ;`
- a. vont afficher 9 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 10 nombres
  - d. vont afficher 11 nombres
17. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
18. L'adresse d'une variable c'est :
- a. le contenu de la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. l'adresse d'un pointeur sur la variable
  - d. ce vers quoi pointe la variable
19. `if (a%2 == 0) printf("bonjour") ;`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand *a* est un entier impair
  - c. Affiche bonjour quand *a* est un entier pair
  - d. N'affiche rien (quelque soit la valeur de *a*)
20. `printf("%c", 'A')`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'

# Sujet n° 181

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", A)`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le code ASCII du caractère 'A'
2. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
  - a. 111011010100110
  - b. 111011010100000
  - c. 111011010101111
  - d. 111011010101000
3. `printf("%c", A)`
  - a. Affiche le caractère 'A'
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
4. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de *a*)
  - b. Affiche bonjour quand *a* est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand *a* est un entier impair
5. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. entraîne l'affichage d'un message d'erreur
  - b. a pour valeur FAUX
  - c. a pour valeur VRAI
  - d. n'a pas de valeur
6. L'expression `a<=1`
  - a. Diminue de 1 la valeur de *a*
  - b. Réalise une affectation
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Utilise les opérateurs `<` et `=`
7. L'adresse d'une variable c'est :
  - a. le numéro de la case mémoire où le contenu de la variable est stocké
  - b. ce vers quoi pointe la variable
  - c. le contenu de la variable
  - d. l'adresse d'un pointeur sur la variable

8. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
9. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- l'adresse de la variable *a*
  - la valeur de *a* tout simplement
  - n'a pas de sens
  - ce vers quoi pointe le pointeur *a*
10. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
11. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
12. `if` est :
- Un mot-clef du langage C
  - Un opérateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un identificateur du langage C
13. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 10
14. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit *a*
  - Augmente la valeur de *a* quelque soit *a*
  - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
15. Après `char c; c='a'; c=c+1;`
- Un message d'erreur s'affiche
  - c* vaut 'b'
  - c* vaut 'A'
  - c* vaut 'a'



16. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Affiche bonjour quand  $a$  est un entier pair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
  - b. ce vers quoi pointe le pointeur `a`
  - c. n'a pas de sens
  - d. l'adresse de la variable `a`
18. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b);`
  - b. `echange(&a, &b);`
  - c. `echange(a++, b++);`
  - d. `echange(*a, *b);`
19. Les instructions
- ```
i=0;
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont afficher 9 nombres
 - b. vont afficher 10 nombres
 - c. vont afficher 11 nombres
 - d. vont boucler indéfiniment
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
 - b. 58
 - c. 68
 - d. 54

Sujet n° 182

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", 'A')`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
2. `printf("%c", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère 'A'
3. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand *a* est un entier pair
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Affiche bonjour quand *a* est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
4. `printf("%i", A)`
 - a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
5. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur FAUX
 - c. a pour valeur VRAI
 - d. n'a pas de valeur
6. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un mot-clef du langage C
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C
7. L'expression `a<=1`
 - a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Réalise une affectation
 - c. Utilise les opérateurs `<` et `=`
 - d. Diminue de 1 la valeur de *a*

8. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b);`
 - `echange(a++, b++);`
 - `echange(*a, *b);`
 - `echange(&a, &b);`
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
12. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
 - l'adresse de la variable `a`
13. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de `a` quelque soit `a`
 - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - Affiche bonjour quelque soit `a`
14. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand `a` est un entier pair
 - Affiche bonjour quand `a` est un entier impair
 - N'affiche rien (quelque soit la valeur de `a`)
15. Si on ajoute 1 au nombre qui se note en base 2 `111011010100111`, on obtient le nombre qui se note en base 2 :
- `111011010101111`
 - `111011010100110`
 - `111011010101000`
 - `111011010100000`

16. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- a. Permettent de d'affecter à `t[10]` la valeur 10
 - b. Permettent de d'affecter à `t[10]` la valeur 45
 - c. Permettent de d'affecter à `t[10]` la valeur 100
 - d. Permettent de d'affecter à `t[10]` la valeur 0
17. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- a. vont afficher 10 nombres
 - b. vont afficher 11 nombres
 - c. vont afficher 9 nombres
 - d. vont boucler indéfiniment
18. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 58
 - b. 62
 - c. 68
 - d. 54
19. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'b'`
 - b. `c` vaut `'A'`
 - c. Un message d'erreur s'affiche
 - d. `c` vaut `'a'`
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. n'a pas de sens

Sujet n° 183

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

2. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le code ASCII du caractère `'A'`
- c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- d. Affiche le caractère `'A'`

3. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101111
- c. 111011010100000
- d. 111011010101000

4. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(*a, *b);`
- b. `echange(a, b);`
- c. `echange(&a, &b);`
- d. `echange(a++, b++);`

5. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. a pour valeur FAUX
- b. n'a pas de valeur
- c. a pour valeur VRAI
- d. entraîne l'affichage d'un message d'erreur

6. Après `char c; c='a'; c=c+1;`

- a. `c` vaut `'a'`
- b. `c` vaut `'b'`
- c. `c` vaut `'A'`
- d. Un message d'erreur s'affiche

7. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 54
 - 58
 - 68
8. L'adresse d'une variable c'est :
- le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
9. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
 - n'a pas de sens
 - la valeur de `a` tout simplement
 - ce vers quoi pointe le pointeur `a`
10. Les instructions `i=0 ;`
- ```
while(i<10)
 printf("%i ", i);
 i++;
```
- vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
  - vont afficher 11 nombres
11. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
12. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
13. `if (a%2 == 0) printf("bonjour") ;`
- Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand `a` est un entier pair
  - Affiche bonjour quand `a` est un entier impair
  - N'affiche rien (quelque soit la valeur de `a`)
14. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - Affiche bonjour quelque soit `a`
  - Augmente la valeur de `a` quelque soit `a`

15. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
16. L'expression `a<=1`
- Réalise une affectation
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
  - Diminue de 1 la valeur de *a*
17. `if` est :
- Un opérateur du langage C
  - Un identificateur du langage C
  - Une commande qu'on tape dans la fenêtre de commande
  - Un mot-clef du langage C
18. `printf("%i", A)`
- Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
19. `if (a%2 == 0) printf("bonjour") ;`
- N'affiche rien (quelque soit la valeur de *a*)
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
  - Affiche bonjour quand *a* est un entier pair
20. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
  - la valeur de *a* tout simplement
  - l'adresse de la variable *a*
  - n'a pas de sens

# Sujet n° 184

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101111
- c. 111011010100110
- d. 111011010101000

2. `if (a%2 == 0) printf("bonjour");`

- a. Déclenche le message d'erreur `invalid lvalue in assignment`
- b. N'affiche rien (quelque soit la valeur de *a*)
- c. Affiche bonjour quand *a* est un entier pair
- d. Affiche bonjour quand *a* est un entier impair

3. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable *a*
- b. ce vers quoi pointe le pointeur *a*
- c. n'a pas de sens
- d. la valeur de *a* tout simplement

4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

5. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 0

6. L'adresse d'une variable c'est :

- a. le numéro de la case mémoire où le contenu de la variable est stocké
- b. le contenu de la variable
- c. l'adresse d'un pointeur sur la variable
- d. ce vers quoi pointe la variable

7. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a, b) ;`
- b. `echange(a++, b++) ;`
- c. `echange(*a, *b) ;`
- d. `echange(&a, &b) ;`

8. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- b. Affiche le code ASCII du caractère stocké dans la variable `A`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère `'A'`

9. `if (a<5) printf("Bonjour") ; a=a+1 ;`

- a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
- b. Augmente la valeur de `a` quelque soit `a`
- c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
- d. Affiche bonjour quelque soit `a`

10. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 58
- b. 68
- c. 54
- d. 62

11. Les instructions `i=0 ;`

```
while(i<10)
 printf("%i ", i) ;
 i++ ;
```

- a. vont boucler indéfiniment
- b. vont afficher 10 nombres
- c. vont afficher 11 nombres
- d. vont afficher 9 nombres

12. Après `char c ; c='a' ; c=c+1 ;`

- a. `c` vaut `'b'`
- b. `c` vaut `'A'`
- c. `c` vaut `'a'`
- d. Un message d'erreur s'affiche

13. `if (a%2 == 0) printf("bonjour") ;`

- a. N'affiche rien (quelque soit la valeur de `a`)
- b. Affiche bonjour quand `a` est un entier pair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. Affiche bonjour quand `a` est un entier impair

14. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un mot-clef du langage C
- c. Un identificateur du langage C
- d. Un opérateur du langage C

15. `printf("%i", 'A')`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le caractère 'A'
  - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère stocké dans la variable *A*
16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
  - b. n'a pas de sens
  - c. ce vers quoi pointe le pointeur `a`
  - d. l'adresse de la variable `a`
17. `printf("%c", 'A')`
- a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
18. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. n'a pas de valeur
  - b. a pour valeur VRAI
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur FAUX
19. L'expression `a<=1`
- a. Diminue de 1 la valeur de *a*
  - b. Réalise une affectation
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Utilise les opérateurs `<` et `=`
20. `printf("%i", A)`
- a. Affiche le caractère 'A'
  - b. Affiche le code ASCII du caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*



# Sujet n° 185

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", 'A')`
  - a. Affiche le code ASCII du caractère stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
2. Après `char c ; c='a' ; c=c+1 ;`
  - a. *c* vaut 'b'
  - b. Un message d'erreur s'affiche
  - c. *c* vaut 'A'
  - d. *c* vaut 'a'
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
  - a. 58
  - b. 54
  - c. 68
  - d. 62
4. `printf("%i", 'A')`
  - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - b. Affiche le caractère 'A'
  - c. Affiche le code ASCII du caractère stocké dans la variable *A*
  - d. Affiche le code ASCII du caractère 'A'
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
  - a. `echange(&a, &b) ;`
  - b. `echange(*a, *b) ;`
  - c. `echange(a++, b++) ;`
  - d. `echange(a, b) ;`
6. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
  - b. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
  - c. Affiche bonjour quelque soit *a*
  - d. Augmente la valeur de *a* quelque soit *a*
7. `if (a%2 == 0) printf("bonjour") ;`
  - a. N'affiche rien (quelque soit la valeur de *a*)
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. Affiche bonjour quand *a* est un entier pair
  - d. Affiche bonjour quand *a* est un entier impair

8. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
  - l'adresse d'un pointeur sur la variable
  - ce vers quoi pointe la variable
  - le contenu de la variable
9. Les instructions `i=0 ;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- vont afficher 11 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 9 nombres
10. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
11. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010100110
 - 111011010100000
 - 111011010101000
13. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - n'a pas de valeur
 - a pour valeur VRAI
 - entraîne l'affichage d'un message d'erreur
14. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
15. `printf("%i", A)`
- Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère dont le code ASCII est stocké dans la variable `A`

16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. N'affiche rien (quelque soit la valeur de a)
 - d. Affiche bonjour quand a est un entier pair
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. n'a pas de sens
18. L'expression `a--`
- a. Réalise une affectation
 - b. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - c. Diminue de 1 la valeur de a
 - d. Utilise les opérateurs `<` et `=`
19. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. l'adresse de la variable `a`
 - b. la valeur de `a` tout simplement
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
20. `if` est :
- a. Un identificateur du langage C
 - b. Un mot-clef du langage C
 - c. Un opérateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande

Sujet n° 186

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`
 - a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le caractère 'A'
2. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. Affiche bonjour et augmente la valeur de a quelque soit a
 - d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
3. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 68
 - c. 54
 - d. 58
4. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
 - a. 111011010100110
 - b. 111011010100000
 - c. 111011010101111
 - d. 111011010101000
5. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
6. L'adresse d'une variable c'est :
 - a. l'adresse d'un pointeur sur la variable
 - b. le contenu de la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. ce vers quoi pointe la variable
7. `if` est :
 - a. Un mot-clef du langage C
 - b. Une commande qu'on tape dans la fenêtre de commande
 - c. Un identificateur du langage C
 - d. Un opérateur du langage C

8. Les instructions `i=0;`
`while(i<10)`
`printf("%i ", i);`
`i++;`
- vont afficher 11 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
 - vont afficher 9 nombres
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - n'a pas de valeur
10. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
11. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
12. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier pair
 - Affiche bonjour quand *a* est un entier impair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
13. `if (a%2 = 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de *a*)
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
14. Après `char c; c='a'; c=c+1;`
- c* vaut 'a'
 - c* vaut 'A'
 - Un message d'erreur s'affiche
 - c* vaut 'b'
15. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'

16. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :

- a. `echange(a++, b++) ;`
- b. `echange(a, b) ;`
- c. `echange(*a, *b) ;`
- d. `echange(&a, &b) ;`

17. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère stocké dans la variable `A`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le caractère `'A'`
- d. Affiche le code ASCII du caractère `'A'`

18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. n'a pas de sens
- b. la valeur de `a` tout simplement
- c. l'adresse de la variable `a`
- d. ce vers quoi pointe le pointeur `a`

19. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Utilise les opérateurs `<` et `=`
- c. Diminue de 1 la valeur de `a`
- d. Réalise une affectation

20. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`

- a. Permettent de d'affecter à `t[10]` la valeur 0
- b. Permettent de d'affecter à `t[10]` la valeur 10
- c. Permettent de d'affecter à `t[10]` la valeur 100
- d. Permettent de d'affecter à `t[10]` la valeur 45

Sujet n° 187

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut `'a'`
 - b. `c` vaut `'b'`
 - c. `c` vaut `'A'`
 - d. Un message d'erreur s'affiche
2. L'adresse d'une variable c'est :
 - a. le contenu de la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le numéro de la case mémoire où le contenu de la variable est stocké
 - d. ce vers quoi pointe la variable
3. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - b. Augmente la valeur de `a` quelque soit `a`
 - c. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - d. Affiche bonjour quelque soit `a`
4. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
 - a. vont afficher 9 nombres
 - b. vont afficher 10 nombres
 - c. vont afficher 11 nombres
 - d. vont boucler indéfiniment
5. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
6. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand `a` est un entier impair
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Déclenche le message d'erreur `invalid lvalue in assignment`
 - d. Affiche bonjour quand `a` est un entier pair

7. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier pair
 - Affiche bonjour quand a est un entier impair
8. `if` est :
- Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
 - Un identificateur du langage C
9. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
 - l'adresse de la variable `a`
 - la valeur de `a` tout simplement
 - n'a pas de sens
10. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Diminue de 1 la valeur de a
11. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le caractère `'A'`
12. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a, b);`
 - `echange(&a, &b);`
 - `echange(a++, b++);`
 - `echange(*a, *b);`
14. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 0

15. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère stocké dans la variable A
16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur FAUX
 - b. entraîne l'affichage d'un message d'erreur
 - c. a pour valeur VRAI
 - d. n'a pas de valeur
17. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
 - b. 111011010100110
 - c. 111011010100000
 - d. 111011010101111
18. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 58
 - b. 68
 - c. 62
 - d. 54
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le code ASCII du caractère stocké dans la variable A
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

Sujet n° 188

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. n'a pas de valeur
 - b. a pour valeur FAUX
 - c. a pour valeur VRAI
 - d. entraîne l'affichage d'un message d'erreur
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(a++, b++) ;`
 - b. `echange(*a, *b) ;`
 - c. `echange(a, b) ;`
 - d. `echange(&a, &b) ;`
3. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. ce vers quoi pointe le pointeur `a`
 - c. n'a pas de sens
 - d. l'adresse de la variable `a`
4. `if (a%2 == 0) printf("bonjour") ;`
 - a. Affiche bonjour quand `a` est un entier impair
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Affiche bonjour quand `a` est un entier pair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
5. `if` est :
 - a. Une commande qu'on tape dans la fenêtre de commande
 - b. Un opérateur du langage C
 - c. Un identificateur du langage C
 - d. Un mot-clef du langage C
6. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
 - a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`

7. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
8. L'adresse d'une variable c'est :
- l'adresse d'un pointeur sur la variable
 - le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - le contenu de la variable
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 58
 - 68
 - 54
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010100110
 - 111011010100000
 - 111011010101000
12. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
13. Les instructions `i=0 ;`
- ```

while(i<10)
 printf("%i ", i);
 i++;

```
- vont afficher 11 nombres
  - vont afficher 9 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
14. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10

15. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
  - b. ce vers quoi pointe le pointeur `a`
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
16. L'expression `a<=1`
- a. Réalise une affectation
  - b. Diminue de 1 la valeur de `a`
  - c. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - d. Utilise les opérateurs `<` et `=`
17. Après `char c ; c='a' ; c=c+1 ;`
- a. `c` vaut `'a'`
  - b. `c` vaut `'A'`
  - c. Un message d'erreur s'affiche
  - d. `c` vaut `'b'`
18. `if (a%2 == 0) printf("bonjour") ;`
- a. N'affiche rien (quelque soit la valeur de `a`)
  - b. Affiche bonjour quand `a` est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand `a` est un entier impair
19. `if (a<5) printf("Bonjour") ; a=a+1 ;`
- a. Affiche bonjour quelque soit `a`
  - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - d. Augmente la valeur de `a` quelque soit `a`
20. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
  - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le caractère `'A'`

# Sujet n° 189

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM**.

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

2. `printf("%i", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le code ASCII du caractère stocké dans la variable *A*
- c. Affiche le caractère 'A'
- d. Affiche le code ASCII du caractère 'A'

3. `if` est :

- a. Un mot-clef du langage C
- b. Une commande qu'on tape dans la fenêtre de commande
- c. Un identificateur du langage C
- d. Un opérateur du langage C

4. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. entraîne l'affichage d'un message d'erreur
- c. a pour valeur VRAI
- d. a pour valeur FAUX

5. Après `char c; c='a'; c=c+1;`

- a. Un message d'erreur s'affiche
- b. *c* vaut 'A'
- c. *c* vaut 'a'
- d. *c* vaut 'b'

6. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. n'a pas de sens
- b. l'adresse de la variable *a*
- c. ce vers quoi pointe le pointeur *a*
- d. la valeur de *a* tout simplement

7. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Affiche bonjour quelque soit *a*
- b. Augmente la valeur de *a* quelque soit *a*
- c. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
- d. Affiche bonjour et augmente la valeur de *a* quelque soit *a*

8. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
9. `if (a%2 == 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier pair
  - N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
10. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 10
  - Permettent de d'affecter à `t[10]` la valeur 0
11. Les instructions `i=0;`  
`while(i<10)`  
`printf("%i ", i);`  
`i++;`
- vont afficher 9 nombres
  - vont afficher 11 nombres
  - vont boucler indéfiniment
  - vont afficher 10 nombres
12. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère 'A'
13. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
14. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur *a*
  - la valeur de *a* tout simplement
  - l'adresse de la variable *a*
  - n'a pas de sens
15. L'expression `a<=1`
- Réalise une affectation
  - Diminue de 1 la valeur de *a*
  - Utilise les opérateurs `<` et `=`
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon



16. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 62
  - b. 68
  - c. 58
  - d. 54
17. L'adresse d'une variable c'est :
- a. le contenu de la variable
  - b. le numéro de la case mémoire où le contenu de la variable est stocké
  - c. ce vers quoi pointe la variable
  - d. l'adresse d'un pointeur sur la variable
18. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- a. 111011010101000
  - b. 111011010100000
  - c. 111011010100110
  - d. 111011010101111
19. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de *a*)
  - b. Affiche bonjour quand *a* est un entier pair
  - c. Affiche bonjour quand *a* est un entier impair
  - d. Déclenche le message d'erreur `invalid lvalue in assignment`
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- a. `echange(*a, *b);`
  - b. `echange(&a, &b);`
  - c. `echange(a++, b++);`
  - d. `echange(a, b);`

# Sujet n° 190

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", 'A')`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

2. `if` est :

- a. Un mot-clef du langage C
- b. Un opérateur du langage C
- c. Une commande qu'on tape dans la fenêtre de commande
- d. Un identificateur du langage C

3. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

4. L'adresse d'une variable c'est :

- a. ce vers quoi pointe la variable
- b. l'adresse d'un pointeur sur la variable
- c. le contenu de la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

5. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 45
- b. Permettent de d'affecter à `t[10]` la valeur 100
- c. Permettent de d'affecter à `t[10]` la valeur 0
- d. Permettent de d'affecter à `t[10]` la valeur 10

6. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
- c. Diminue de 1 la valeur de *a*
- d. Réalise une affectation

7. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- c. Affiche le code ASCII du caractère stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

8. `if (a<5) printf("Bonjour"); a=a+1;`
- Affiche bonjour quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Augmente la valeur de  $a$  quelque soit  $a$
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur VRAI
  - entraîne l'affichage d'un message d'erreur
  - a pour valeur FAUX
10. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère 'A'
11. On suppose que  $a$  a été déclarée par `int a`. L'expression `*a` a pour valeur :
- n'a pas de sens
  - ce vers quoi pointe le pointeur  $a$
  - l'adresse de la variable  $a$
  - la valeur de  $a$  tout simplement
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010101111
  - 111011010100000
  - 111011010101000
13. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
- `echange(&a, &b);`
  - `echange(a++, b++);`
  - `echange(a, b);`
  - `echange(*a, *b);`
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
  - 68
  - 58
  - 54
15. Après `char c; c='a'; c=c+1;`
- $c$  vaut 'b'
  - $c$  vaut 'a'
  - $c$  vaut 'A'
  - Un message d'erreur s'affiche

16. `if (a%2 == 0) printf("bonjour");`
- a. N'affiche rien (quelque soit la valeur de  $a$ )
  - b. Affiche bonjour quand  $a$  est un entier impair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier pair
17. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. ce vers quoi pointe le pointeur `a`
  - b. n'a pas de sens
  - c. la valeur de `a` tout simplement
  - d. l'adresse de la variable `a`
18. `if (a%2 = 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier impair
  - b. Affiche bonjour quand  $a$  est un entier pair
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. N'affiche rien (quelque soit la valeur de  $a$ )
19. `printf("%i", A)`
- a. Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère `'A'`
  - d. Affiche le code ASCII du caractère stocké dans la variable  $A$
20. Les instructions `i=0;`
- ```
while(i<10)
    printf("%i ", i);
    i++;
```
- a. vont afficher 10 nombres
 - b. vont afficher 11 nombres
 - c. vont afficher 9 nombres
 - d. vont boucler indéfiniment

Sujet n° 191

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```


Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%c", A)`

- a. Affiche le code ASCII du caractère stocké dans la variable *A*
- b. Affiche le caractère 'A'
- c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
- d. Affiche le code ASCII du caractère 'A'

2. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 11 nombres
- b. vont afficher 9 nombres
- c. vont afficher 10 nombres
- d. vont boucler indéfiniment

3. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`

4. On suppose que *a* a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable *a*
- b. ce vers quoi pointe le pointeur *a*
- c. n'a pas de sens
- d. la valeur de *a* tout simplement

5. L'expression `a<=1`

- a. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- b. Utilise les opérateurs `<` et `=`
- c. Diminue de 1 la valeur de *a*
- d. Réalise une affectation

6. On suppose que *a* a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur *a*
- b. la valeur de *a* tout simplement
- c. l'adresse de la variable *a*
- d. n'a pas de sens

7. `printf("%c", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
8. `if (a%2 == 0) printf("bonjour");`
- Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
 - Affiche bonjour quand *a* est un entier pair
 - N'affiche rien (quelque soit la valeur de *a*)
9. Après `char c; c='a'; c=c+1;`
- c* vaut 'A'
 - c* vaut 'b'
 - c* vaut 'a'
 - Un message d'erreur s'affiche
10. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - Affiche bonjour quelque soit *a*
 - N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
11. `if (a%2 = 0) printf("bonjour");`
- Affiche bonjour quand *a* est un entier pair
 - N'affiche rien (quelque soit la valeur de *a*)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand *a* est un entier impair
12. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*
13. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101111
 - 111011010100110
 - 111011010101000
 - 111011010100000
14. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'

15. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 54
 - 62
 - 68
16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a pour valeur VRAI
 - n'a pas de valeur
 - a pour valeur FAUX
 - entraîne l'affichage d'un message d'erreur
17. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
18. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
19. En cours, on a vu comment à l'aide de pointeurs définir une fonction **echange** qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables **a** et **b** on doit écrire :
- `echange(a, b) ;`
 - `echange(&a, &b) ;`
 - `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
20. **if** est :
- Une commande qu'on tape dans la fenêtre de commande
 - Un identificateur du langage C
 - Un mot-clef du langage C
 - Un opérateur du langage C

Sujet n° 192

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `if (a<5) printf("Bonjour") ; a=a+1 ;`
 - a. Affiche bonjour quelque soit a
 - b. Augmente la valeur de a quelque soit a
 - c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - d. Affiche bonjour et augmente la valeur de a quelque soit a
2. L'adresse d'une variable c'est :
 - a. le numéro de la case mémoire où le contenu de la variable est stocké
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. ce vers quoi pointe la variable
3. L'expression `a<=1`
 - a. Réalise une affectation
 - b. Utilise les opérateurs `<` et `=`
 - c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - d. Diminue de 1 la valeur de a
4. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 54
 - b. 62
 - c. 68
 - d. 58
5. `printf("%i", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère `'A'`
6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. l'adresse de la variable `a`
 - b. n'a pas de sens
 - c. la valeur de `a` tout simplement
 - d. ce vers quoi pointe le pointeur `a`
7. `printf("%c", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - d. Affiche le caractère `'A'`

8. `if` est :
- Un identificateur du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un mot-clef du langage C
 - Un opérateur du langage C
9. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 9 nombres
 - vont afficher 10 nombres
 - vont boucler indéfiniment
 - vont afficher 11 nombres
10. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand a est un entier impair
 - Affiche bonjour quand a est un entier pair
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - N'affiche rien (quelque soit la valeur de a)
11. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
 - 111011010101111
 - 111011010101000
 - 111011010100000
12. Après `char c ; c='a' ; c=c+1 ;`
- c vaut 'b'
 - Un message d'erreur s'affiche
 - c vaut 'a'
 - c vaut 'A'
13. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand a est un entier pair
 - N'affiche rien (quelque soit la valeur de a)
 - Déclenche le message d'erreur `invalid lvalue in assignment`
 - Affiche bonjour quand a est un entier impair
14. `printf("%i", A)`
- Affiche le code ASCII du caractère stocké dans la variable A
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable A
 - Affiche le caractère 'A'
15. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à $t[10]$ la valeur 0
 - Permettent de d'affecter à $t[10]$ la valeur 100
 - Permettent de d'affecter à $t[10]$ la valeur 10
 - Permettent de d'affecter à $t[10]$ la valeur 45

16. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`
17. `printf("%c", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable `A`
 - b. Affiche le code ASCII du caractère `'A'`
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - d. Affiche le caractère `'A'`
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
19. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. a pour valeur VRAI
 - b. n'a pas de valeur
 - c. a pour valeur FAUX
 - d. entraîne l'affichage d'un message d'erreur
20. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b);`
 - b. `echange(*a, *b);`
 - c. `echange(a++, b++);`
 - d. `echange(&a, &b);`

Sujet n° 193

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
 - a. la valeur de `a` tout simplement
 - b. l'adresse de la variable `a`
 - c. n'a pas de sens
 - d. ce vers quoi pointe le pointeur `a`
2. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(*a, *b) ;`
 - b. `echange(a, b) ;`
 - c. `echange(a++, b++) ;`
 - d. `echange(&a, &b) ;`
3. `printf("%i", A)`
 - a. Affiche le code ASCII du caractère '`A`'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - c. Affiche le caractère '`A`'
 - d. Affiche le code ASCII du caractère stocké dans la variable `A`
4. Après `char c ; c='a' ; c=c+1 ;`
 - a. `c` vaut '`A`'
 - b. `c` vaut '`b`'
 - c. Un message d'erreur s'affiche
 - d. `c` vaut '`a`'
5. L'expression `a<=1`
 - a. `A` pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - b. Utilise les opérateurs `<` et `=`
 - c. Réalise une affectation
 - d. Diminue de 1 la valeur de `a`
6. `if` est :
 - a. Un opérateur du langage C
 - b. Un identificateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un mot-clef du langage C

7. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

8. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier pair
- b. Affiche bonjour quand a est un entier impair
- c. N'affiche rien (quelque soit la valeur de a)
- d. Déclenche le message d'erreur `invalid lvalue in assignment`

9. L'adresse d'une variable c'est :

- a. le contenu de la variable
- b. ce vers quoi pointe la variable
- c. l'adresse d'un pointeur sur la variable
- d. le numéro de la case mémoire où le contenu de la variable est stocké

10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010101000
- b. 111011010101111
- c. 111011010100110
- d. 111011010100000

11. `printf("%c", A)`

- a. Affiche le caractère dont le code ASCII est stocké dans la variable A
- b. Affiche le code ASCII du caractère stocké dans la variable A
- c. Affiche le caractère 'A'
- d. Affiche le code ASCII du caractère 'A'

12. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. la valeur de a tout simplement
- b. n'a pas de sens
- c. ce vers quoi pointe le pointeur a
- d. l'adresse de la variable a

13. Les instructions `i=0;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont afficher 10 nombres
- b. vont afficher 9 nombres
- c. vont boucler indéfiniment
- d. vont afficher 11 nombres

14. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable A
- c. Affiche le caractère 'A'
- d. Affiche le code ASCII du caractère stocké dans la variable A

15. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. N'affiche rien (quelque soit la valeur de a)
 - c. Affiche bonjour quand a est un entier pair
 - d. Affiche bonjour quand a est un entier impair
16. L'expression `(0 == 7%3) || (1 == 9%3)`
- a. entraîne l'affichage d'un message d'erreur
 - b. a pour valeur FAUX
 - c. n'a pas de valeur
 - d. a pour valeur VRAI
17. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à `t[10]` la valeur 45
 - b. Permettent de d'affecter à `t[10]` la valeur 100
 - c. Permettent de d'affecter à `t[10]` la valeur 0
 - d. Permettent de d'affecter à `t[10]` la valeur 10
18. `printf("%i", 'A')`
- a. Affiche le caractère 'A'
 - b. Affiche le code ASCII du caractère stocké dans la variable A
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
19. `if (a<5) printf("Bonjour"); a=a+1;`
- a. Augmente la valeur de a quelque soit a
 - b. Affiche bonjour et augmente la valeur de a quelque soit a
 - c. Affiche bonjour quelque soit a
 - d. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
20. Le nombre qui se note 110110 en base 2 se note en base 10 :
- a. 54
 - b. 58
 - c. 62
 - d. 68

Sujet n° 194

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Le nombre qui se note 110110 en base 2 se note en base 10 :

- a. 62
- b. 58
- c. 54
- d. 68

2. Les instructions `i=0 ;`

```
while(i<10)
    printf("%i ", i);
    i++;
```

- a. vont boucler indéfiniment
- b. vont afficher 9 nombres
- c. vont afficher 10 nombres
- d. vont afficher 11 nombres

3. `printf("%c", A)`

- a. Affiche le caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable A
- c. Affiche le caractère dont le code ASCII est stocké dans la variable A
- d. Affiche le code ASCII du caractère 'A'

4. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. l'adresse de la variable `a`
- b. n'a pas de sens
- c. ce vers quoi pointe le pointeur `a`
- d. la valeur de `a` tout simplement

5. `printf("%c", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère 'A'
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

6. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable `a`
- b. la valeur de `a` tout simplement
- c. n'a pas de sens
- d. ce vers quoi pointe le pointeur `a`

7. `printf("%i", 'A')`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le code ASCII du caractère stocké dans la variable A
- c. Affiche le caractère 'A'
- d. Affiche le caractère dont le code ASCII est stocké dans la variable A

8. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de a quelque soit a
 - N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
 - Affiche bonjour et augmente la valeur de a quelque soit a
 - Affiche bonjour quelque soit a
9. L'expression `a<=1`
- Utilise les opérateurs `<` et `=`
 - Réalise une affectation
 - Diminue de 1 la valeur de a
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
10. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010100110
 - 111011010101111
 - 111011010101000
11. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - a pour valeur VRAI
 - n'a pas de valeur
 - a pour valeur FAUX
12. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
13. Après `char c; c='a'; c=c+1;`
- c vaut 'a'
 - c vaut 'A'
 - Un message d'erreur s'affiche
 - c vaut 'b'
14. `if` est :
- Un identificateur du langage C
 - Un mot-clef du langage C
 - Une commande qu'on tape dans la fenêtre de commande
 - Un opérateur du langage C
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(a++, b++) ;`
 - `echange(*a, *b) ;`
 - `echange(a, b) ;`
 - `echange(&a, &b) ;`

16. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand a est un entier impair
 - b. Déclenche le message d'erreur `invalid lvalue in assignment`
 - c. Affiche bonjour quand a est un entier pair
 - d. N'affiche rien (quelque soit la valeur de a)
17. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
 - b. l'adresse d'un pointeur sur la variable
 - c. le contenu de la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
18. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand a est un entier impair
 - c. Affiche bonjour quand a est un entier pair
 - d. N'affiche rien (quelque soit la valeur de a)
19. `printf("%i", A)`
- a. Affiche le code ASCII du caractère stocké dans la variable A
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
20. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- a. Permettent de d'affecter à $t[10]$ la valeur 0
 - b. Permettent de d'affecter à $t[10]$ la valeur 100
 - c. Permettent de d'affecter à $t[10]$ la valeur 45
 - d. Permettent de d'affecter à $t[10]$ la valeur 10

Sujet n° 195

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'expression `(0 == 7%3) || (1 == 9%3)`
 - a. n'a pas de valeur
 - b. a pour valeur VRAI
 - c. a pour valeur FAUX
 - d. entraîne l'affichage d'un message d'erreur
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
 - a. n'a pas de sens
 - b. la valeur de `a` tout simplement
 - c. ce vers quoi pointe le pointeur `a`
 - d. l'adresse de la variable `a`
3. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
 - b. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
 - c. Affiche bonjour quelque soit `a`
 - d. Augmente la valeur de `a` quelque soit `a`
4. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand `a` est un entier pair
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Affiche bonjour quand `a` est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`
5. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
 - a. `echange(a, b);`
 - b. `echange(*a, *b);`
 - c. `echange(a++, b++);`
 - d. `echange(&a, &b);`
6. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand `a` est un entier pair
 - b. N'affiche rien (quelque soit la valeur de `a`)
 - c. Affiche bonjour quand `a` est un entier impair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`

7. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
 - 111011010101111
 - 111011010100110
 - 111011010101000
8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
 - 62
 - 54
 - 68
9. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
10. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - l'adresse d'un pointeur sur la variable
 - ce vers quoi pointe la variable
 - le contenu de la variable
11. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
12. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 10
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
13. L'expression `a<=1`
- Diminue de 1 la valeur de *a*
 - A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
14. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère stocké dans la variable *A*

15. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- a. vont afficher 11 nombres
 - b. vont boucler indéfiniment
 - c. vont afficher 10 nombres
 - d. vont afficher 9 nombres
16. `if` est :
- a. Un mot-clef du langage C
 - b. Un identificateur du langage C
 - c. Une commande qu'on tape dans la fenêtre de commande
 - d. Un opérateur du langage C
17. `printf("%i", A)`
- a. Affiche le caractère 'A'
 - b. Affiche le caractère dont le code ASCII est stocké dans la variable A
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le code ASCII du caractère 'A'
18. `printf("%c", 'A')`
- a. Affiche le code ASCII du caractère 'A'
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère stocké dans la variable A
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable A
19. Après `char c ; c='a' ; c=c+1 ;`
- a. c vaut 'b'
 - b. c vaut 'a'
 - c. Un message d'erreur s'affiche
 - d. c vaut 'A'
20. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. l'adresse de la variable `a`
 - b. ce vers quoi pointe le pointeur `a`
 - c. la valeur de `a` tout simplement
 - d. n'a pas de sens

Sujet n° 196

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. `printf("%i", 'A')`
 - a. Affiche le code ASCII du caractère stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère 'A'
 - c. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - d. Affiche le caractère 'A'
2. Le nombre qui se note 110110 en base 2 se note en base 10 :
 - a. 62
 - b. 68
 - c. 58
 - d. 54
3. `printf("%i", A)`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le code ASCII du caractère stocké dans la variable *A*
 - c. Affiche le caractère 'A'
 - d. Affiche le code ASCII du caractère 'A'
4. `if (a<5) printf("Bonjour"); a=a+1;`
 - a. Affiche bonjour quelque soit *a*
 - b. N'affiche pas bonjour et n'augmente pas la valeur de *a* quelque soit *a*
 - c. Affiche bonjour et augmente la valeur de *a* quelque soit *a*
 - d. Augmente la valeur de *a* quelque soit *a*
5. `printf("%c", 'A')`
 - a. Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - b. Affiche le caractère 'A'
 - c. Affiche le code ASCII du caractère 'A'
 - d. Affiche le code ASCII du caractère stocké dans la variable *A*
6. `if (a%2 == 0) printf("bonjour");`
 - a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier pair
 - c. Affiche bonjour quand *a* est un entier impair
 - d. N'affiche rien (quelque soit la valeur de *a*)
7. `if (a%2 == 0) printf("bonjour");`
 - a. Affiche bonjour quand *a* est un entier impair
 - b. N'affiche rien (quelque soit la valeur de *a*)
 - c. Affiche bonjour quand *a* est un entier pair
 - d. Déclenche le message d'erreur `invalid lvalue in assignment`

8. Les instructions `i=0 ;`
`while(i<10)`
`printf("%i ", i) ;`
`i++ ;`
- vont afficher 11 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres
9. Les instructions `t[0] = 0 ; for(i=1 ; i<=10 ; i++) t[i] = t[i-1]+i ;`
- Permettent de d'affecter à `t[10]` la valeur 45
 - Permettent de d'affecter à `t[10]` la valeur 100
 - Permettent de d'affecter à `t[10]` la valeur 0
 - Permettent de d'affecter à `t[10]` la valeur 10
10. L'adresse d'une variable c'est :
- le numéro de la case mémoire où le contenu de la variable est stocké
 - ce vers quoi pointe la variable
 - le contenu de la variable
 - l'adresse d'un pointeur sur la variable
11. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
 - `c` vaut `'a'`
 - `c` vaut `'A'`
 - `c` vaut `'b'`
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010101000
 - 111011010100110
 - 111011010100000
 - 111011010101111
13. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- la valeur de `a` tout simplement
 - l'adresse de la variable `a`
 - ce vers quoi pointe le pointeur `a`
 - n'a pas de sens
14. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de `a`
 - Réalise une affectation
15. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur VRAI
 - a pour valeur FAUX

16. `if` est :
- a. Un mot-clef du langage C
 - b. Un identificateur du langage C
 - c. Un opérateur du langage C
 - d. Une commande qu'on tape dans la fenêtre de commande
17. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- a. `echange(a, b) ;`
 - b. `echange(a++, b++) ;`
 - c. `echange(&a, &b) ;`
 - d. `echange(*a, *b) ;`
18. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- a. n'a pas de sens
 - b. ce vers quoi pointe le pointeur `a`
 - c. l'adresse de la variable `a`
 - d. la valeur de `a` tout simplement
19. `printf("%c", A)`
- a. Affiche le code ASCII du caractère '`A`'
 - b. Affiche le caractère '`A`'
 - c. Affiche le code ASCII du caractère stocké dans la variable `A`
 - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
20. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
 - b. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
 - c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
 - d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

Sujet n° 197

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100110
- b. 111011010101111
- c. 111011010100000
- d. 111011010101000

2. `if (a%2 == 0) printf("bonjour");`

- a. Affiche bonjour quand a est un entier pair
- b. Affiche bonjour quand a est un entier impair
- c. Déclenche le message d'erreur `invalid lvalue in assignment`
- d. N'affiche rien (quelque soit la valeur de a)

3. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`

4. On suppose que a a été déclarée par `int a`. L'expression `*a` a pour valeur :

- a. l'adresse de la variable a
- b. ce vers quoi pointe le pointeur a
- c. la valeur de a tout simplement
- d. n'a pas de sens

5. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 10
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 100
- d. Permettent de d'affecter à `t[10]` la valeur 0

6. `if (a<5) printf("Bonjour"); a=a+1;`

- a. Augmente la valeur de a quelque soit a
- b. Affiche bonjour et augmente la valeur de a quelque soit a
- c. N'affiche pas bonjour et n'augmente pas la valeur de a quelque soit a
- d. Affiche bonjour quelque soit a

7. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un mot-clef du langage C
- c. Un opérateur du langage C
- d. Un identificateur du langage C

8. Après `char c ; c='a' ; c=c+1 ;`
- `c` vaut `'a'`
 - `c` vaut `'b'`
 - `c` vaut `'A'`
 - Un message d'erreur s'affiche
9. L'expression `(0 == 7%3) || (1 == 9%3)`
- entraîne l'affichage d'un message d'erreur
 - n'a pas de valeur
 - a pour valeur VRAI
 - a pour valeur FAUX
10. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
- `echange(*a, *b) ;`
 - `echange(a++, b++) ;`
 - `echange(&a, &b) ;`
 - `echange(a, b) ;`
11. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 68
 - 54
 - 58
 - 62
12. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
13. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
14. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- l'adresse de la variable `a`
 - n'a pas de sens
 - ce vers quoi pointe le pointeur `a`
 - la valeur de `a` tout simplement
15. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
 - Affiche le code ASCII du caractère `'A'`
 - Affiche le caractère `'A'`
 - Affiche le code ASCII du caractère stocké dans la variable `A`

16. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
 - b. le contenu de la variable
 - c. l'adresse d'un pointeur sur la variable
 - d. le numéro de la case mémoire où le contenu de la variable est stocké
17. L'expression `a<=1`
- a. Utilise les opérateurs `<` et `=`
 - b. Réalise une affectation
 - c. Diminue de 1 la valeur de *a*
 - d. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
18. `if (a%2 == 0) printf("bonjour");`
- a. Déclenche le message d'erreur `invalid lvalue in assignment`
 - b. Affiche bonjour quand *a* est un entier pair
 - c. N'affiche rien (quelque soit la valeur de *a*)
 - d. Affiche bonjour quand *a* est un entier impair
19. Les instructions `i=0;`
- ```
while(i<10)
 printf("%i ", i);
 i++;
```
- a. vont afficher 9 nombres
  - b. vont boucler indéfiniment
  - c. vont afficher 11 nombres
  - d. vont afficher 10 nombres
20. `printf("%c", A)`
- a. Affiche le code ASCII du caractère 'A'
  - b. Affiche le code ASCII du caractère stocké dans la variable *A*
  - c. Affiche le caractère 'A'
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable *A*

# Sujet n° 198

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
  - a. le contenu de la variable
  - b. ce vers quoi pointe la variable
  - c. l'adresse d'un pointeur sur la variable
  - d. le numéro de la case mémoire où le contenu de la variable est stocké
2. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
  - a. la valeur de `a` tout simplement
  - b. ce vers quoi pointe le pointeur `a`
  - c. l'adresse de la variable `a`
  - d. n'a pas de sens
3. Après `char c ; c='a' ; c=c+1 ;`
  - a. `c` vaut `'b'`
  - b. Un message d'erreur s'affiche
  - c. `c` vaut `'a'`
  - d. `c` vaut `'A'`
4. Les instructions 

```
i=0 ;
while(i<10)
 printf("%i ", i) ;
 i++ ;
```

  - a. vont boucler indéfiniment
  - b. vont afficher 10 nombres
  - c. vont afficher 9 nombres
  - d. vont afficher 11 nombres
5. `if` est :
  - a. Un mot-clef du langage C
  - b. Un identificateur du langage C
  - c. Une commande qu'on tape dans la fenêtre de commande
  - d. Un opérateur du langage C
6. `printf("%c", A)`
  - a. Affiche le code ASCII du caractère `'A'`
  - b. Affiche le caractère `'A'`
  - c. Affiche le code ASCII du caractère stocké dans la variable `A`
  - d. Affiche le caractère dont le code ASCII est stocké dans la variable `A`

7. `if (a<5) printf("Bonjour"); a=a+1;`
- Augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
  - Affiche bonjour quelque soit  $a$
  - N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
8. `printf("%i", 'A')`
- Affiche le code ASCII du caractère stocké dans la variable  $A$
  - Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable  $A$
9. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables  $a$  et  $b$  on doit écrire :
- `echange(*a, *b);`
  - `echange(&a, &b);`
  - `echange(a++, b++);`
  - `echange(a, b);`
10. On suppose que  $a$  a été déclarée par `int a`. L'expression `*a` a pour valeur :
- ce vers quoi pointe le pointeur  $a$
  - n'a pas de sens
  - la valeur de  $a$  tout simplement
  - l'adresse de la variable  $a$
11. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de  $a$ )
  - Affiche bonjour quand  $a$  est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand  $a$  est un entier impair
12. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100000
  - 111011010101000
  - 111011010101111
  - 111011010100110
13. L'expression `(0 == 7%3) || (1 == 9%3)`
- n'a pas de valeur
  - a pour valeur FAUX
  - a pour valeur VRAI
  - entraîne l'affichage d'un message d'erreur
14. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 58
  - 62
  - 54
  - 68

15. `printf("%c", 'A')`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le code ASCII du caractère stocké dans la variable *A*
16. L'expression `a<=1`
- Diminue de 1 la valeur de *a*
  - A pour valeur VRAI si  $a \leq 1$  et FAUX sinon
  - Utilise les opérateurs `<` et `=`
  - Réalise une affectation
17. `if (a%2 == 0) printf("bonjour");`
- N'affiche rien (quelque soit la valeur de *a*)
  - Affiche bonjour quand *a* est un entier impair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier pair
18. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?
- `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
  - `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`
  - `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
19. `printf("%i", A)`
- Affiche le code ASCII du caractère 'A'
  - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
  - Affiche le caractère 'A'
  - Affiche le code ASCII du caractère stocké dans la variable *A*
20. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`
- Permettent de d'affecter à `t[10]` la valeur 45
  - Permettent de d'affecter à `t[10]` la valeur 100
  - Permettent de d'affecter à `t[10]` la valeur 0
  - Permettent de d'affecter à `t[10]` la valeur 10

# Sujet n° 199

Langage C  
1<sup>er</sup> semestre

Examen Février 2008  
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

## EXERCICE 1.

Écrire un programme qui stocke dans un tableau les  $n$  premiers termes de la suite de Fibonacci. Le nombre  $n$  est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par  $u_0 = u_1 = 1$  et par  $u_n = u_{n-2} + u_{n-1}$  pour tout  $n \geq 2$ .

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

## EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants :  $[0, 10[$ ,  $[10, 12[$ ,  $[12, 14[$ ,  $[14, 16[$ ,  $[16, 20]$ .

## EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers  $a$  et  $b$  et qui affiche un rectangle comprenant  $b$  lignes, chacune constituée de  $a$  fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

## Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

 int i, j;
 int t[MAX];

 i=0;//Point d'observation 1

 while(i<MAX){
 i++; //Point d'observation 2
 t[i] = i;
 }

 j=1; //Point d'observation 3

 do{
 t[j] = t[j-1]+t[j];
 j++; //Point d'observation 4
 }
 while(j<=MAX-1);

 //Point d'observation 5

 printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```



## Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. L'adresse d'une variable c'est :
  - a. l'adresse d'un pointeur sur la variable
  - b. ce vers quoi pointe la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. le contenu de la variable
2. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
  - a. l'adresse de la variable `a`
  - b. la valeur de `a` tout simplement
  - c. n'a pas de sens
  - d. ce vers quoi pointe le pointeur `a`
3. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables `a` et `b` on doit écrire :
  - a. `echange(a++, b++) ;`
  - b. `echange(a, b) ;`
  - c. `echange(*a, *b) ;`
  - d. `echange(&a, &b) ;`
4. L'expression `(0 == 7%3) || (1 == 9%3)`
  - a. a pour valeur FAUX
  - b. n'a pas de valeur
  - c. entraîne l'affichage d'un message d'erreur
  - d. a pour valeur VRAI
5. `if (a%2 == 0) printf("bonjour") ;`
  - a. Déclenche le message d'erreur `invalid lvalue in assignment`
  - b. Affiche bonjour quand `a` est un entier pair
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier impair
6. `if (a%2 == 0) printf("bonjour") ;`
  - a. Affiche bonjour quand `a` est un entier pair
  - b. Déclenche le message d'erreur `invalid lvalue in assignment`
  - c. N'affiche rien (quelque soit la valeur de `a`)
  - d. Affiche bonjour quand `a` est un entier impair
7. `if (a<5) printf("Bonjour") ; a=a+1 ;`
  - a. Augmente la valeur de `a` quelque soit `a`
  - b. N'affiche pas bonjour et n'augmente pas la valeur de `a` quelque soit `a`
  - c. Affiche bonjour et augmente la valeur de `a` quelque soit `a`
  - d. Affiche bonjour quelque soit `a`

8. Après `char c ; c='a' ; c=c+1 ;`
- Un message d'erreur s'affiche
  - `c` vaut `'b'`
  - `c` vaut `'A'`
  - `c` vaut `'a'`
9. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :
- 111011010100110
  - 111011010101000
  - 111011010101111
  - 111011010100000
10. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 54
  - 62
  - 68
  - 58
11. `printf("%i", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
12. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :
- ce vers quoi pointe le pointeur `a`
  - n'a pas de sens
  - la valeur de `a` tout simplement
  - l'adresse de la variable `a`
13. `printf("%c", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le caractère `'A'`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
14. `printf("%c", 'A')`
- Affiche le caractère dont le code ASCII est stocké dans la variable `A`
  - Affiche le code ASCII du caractère stocké dans la variable `A`
  - Affiche le code ASCII du caractère `'A'`
  - Affiche le caractère `'A'`
15. Les instructions
- ```
i=0 ;
while(i<10)
    printf("%i ", i) ;
    i++ ;
```
- vont afficher 11 nombres
 - vont afficher 9 nombres
 - vont boucler indéfiniment
 - vont afficher 10 nombres

16. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- c. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

17. `printf("%i", A)`

- a. Affiche le code ASCII du caractère 'A'
- b. Affiche le caractère dont le code ASCII est stocké dans la variable A
- c. Affiche le code ASCII du caractère stocké dans la variable A
- d. Affiche le caractère 'A'

18. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 10
- b. Permettent de d'affecter à `t[10]` la valeur 45
- c. Permettent de d'affecter à `t[10]` la valeur 100
- d. Permettent de d'affecter à `t[10]` la valeur 0

19. L'expression `a<=1`

- a. Utilise les opérateurs `<` et `=`
- b. Réalise une affectation
- c. A pour valeur VRAI si $a \leq 1$ et FAUX sinon
- d. Diminue de 1 la valeur de a

20. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un identificateur du langage C
- c. Un mot-clef du langage C
- d. Un opérateur du langage C

Sujet n° 200

Langage C
1^{er} semestre

Examen Février 2008
Documents et calculatrices interdits

ATTENTION : pour le QCM et le traçage, rendre exclusivement la feuille réponse jointe au sujet, que vous glisserez dans votre copie. Pour limiter le risque de perte, pour préserver l'anonymat et pour faciliter la correction : il est **IMPÉRATIF** de **recopier le numéro du sujet sur votre copie, ET sur la feuille réponse du QCM.**

EXERCICE 1.

Écrire un programme qui stocke dans un tableau les n premiers termes de la suite de Fibonacci. Le nombre n est laissé au choix de l'utilisateur. On rappelle que cette suite est définie par $u_0 = u_1 = 1$ et par $u_n = u_{n-2} + u_{n-1}$ pour tout $n \geq 2$.

Remarque : ne pas tenir compte des problèmes de dépassement des valeurs maximales autorisées pour les types de donnée du langage C.

EXERCICE 2.

Écrire une fonction `mention` qui prend en entrée une note (potentiellement un nombre à virgule) et qui affiche la mention associée à cette note. C'est-à-dire que la mention affichée doit être "Recalé", "Passable", "AB", "B" ou "TB" selon que la note se situe respectivement dans les intervalles suivants : $[0, 10[$, $[10, 12[$, $[12, 14[$, $[14, 16[$, $[16, 20]$.

EXERCICE 3.

Écrire une fonction `rectangle` qui prend en entrée deux entiers a et b et qui affiche un rectangle comprenant b lignes, chacune constituée de a fois la caractère 'x'. La fonction devra retourner le nombre de 'x' affichés.

Exercice 4

Attention : pour cet exercice, utiliser uniquement la feuille réponse prévue à cet effet.

1. Tracer le programme ci-dessous (utiliser le tableau pré-rempli dans la feuille réponse).
2. Quel message s'affiche à la fin de l'exécution ?
3. Expliquer ce que fait le programme en fonction de la constante `MAX` définie dans le préprocesseur.

```
#include<stdio.h>
#define MAX 4

main(){

    int i, j;
    int t[MAX];

    i=0;//Point d'observation 1

    while(i<MAX){
        i++; //Point d'observation 2
        t[i] = i;
    }

    j=1; //Point d'observation 3

    do{
        t[j] = t[j-1]+t[j];
        j++; //Point d'observation 4
    }
    while(j<=MAX-1);

    //Point d'observation 5

    printf("t[%i] vaut %i\n", MAX-1, t[MAX-1]);
}
```

Exercice 5

QCM. Rappel important : voir en début de sujet le mode d'emploi pour répondre au QCM

1. Si on ajoute 1 au nombre qui se note en base 2 111011010100111, on obtient le nombre qui se note en base 2 :

- a. 111011010100000
- b. 111011010101111
- c. 111011010100110
- d. 111011010101000

2. On suppose que `a` a été déclarée par `int a`. L'expression `&a` a pour valeur :

- a. ce vers quoi pointe le pointeur `a`
- b. n'a pas de sens
- c. la valeur de `a` tout simplement
- d. l'adresse de la variable `a`

3. `if` est :

- a. Une commande qu'on tape dans la fenêtre de commande
- b. Un opérateur du langage C
- c. Un identificateur du langage C
- d. Un mot-clef du langage C

4. Laquelle des quatre fonctions suivantes permet-elle d'échanger les valeurs de deux variables entières ?

- a. `void echange(int &a, int &b) {int t; t=&a; &a=&b; &b=t;}`
- b. `void echange(int *a, int *b) {int t; t=&a; &a=&b; &b=t;}`
- c. `void echange(int &a, int &b) {int t; t=*a; *a=*b; *b=t;}`
- d. `void echange(int *a, int *b) {int t; t=*a; *a=*b; *b=t;}`

5. `printf("%c", 'A')`

- a. Affiche le caractère `'A'`
- b. Affiche le caractère dont le code ASCII est stocké dans la variable `A`
- c. Affiche le code ASCII du caractère `'A'`
- d. Affiche le code ASCII du caractère stocké dans la variable `A`

6. L'expression `(0 == 7%3) || (1 == 9%3)`

- a. n'a pas de valeur
- b. a pour valeur VRAI
- c. entraîne l'affichage d'un message d'erreur
- d. a pour valeur FAUX

7. Les instructions `t[0] = 0; for(i=1; i<=10; i++) t[i] = t[i-1]+i;`

- a. Permettent de d'affecter à `t[10]` la valeur 100
- b. Permettent de d'affecter à `t[10]` la valeur 0
- c. Permettent de d'affecter à `t[10]` la valeur 10
- d. Permettent de d'affecter à `t[10]` la valeur 45

8. Le nombre qui se note 110110 en base 2 se note en base 10 :
- 62
 - 58
 - 54
 - 68
9. `printf("%i", 'A')`
- Affiche le code ASCII du caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
10. `printf("%i", A)`
- Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le code ASCII du caractère 'A'
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
11. L'expression `a<=1`
- A pour valeur VRAI si $a \leq 1$ et FAUX sinon
 - Réalise une affectation
 - Utilise les opérateurs `<` et `=`
 - Diminue de 1 la valeur de *a*
12. `printf("%c", A)`
- Affiche le code ASCII du caractère 'A'
 - Affiche le caractère dont le code ASCII est stocké dans la variable *A*
 - Affiche le caractère 'A'
 - Affiche le code ASCII du caractère stocké dans la variable *A*
13. Les instructions `i=0 ;`
- ```

while(i<10)
 printf("%i ", i) ;
 i++ ;

```
- vont boucler indéfiniment
  - vont afficher 11 nombres
  - vont afficher 10 nombres
  - vont afficher 9 nombres
14. `if (a%2 == 0) printf("bonjour") ;`
- Affiche bonjour quand *a* est un entier pair
  - Déclenche le message d'erreur `invalid lvalue in assignment`
  - Affiche bonjour quand *a* est un entier impair
  - N'affiche rien (quelque soit la valeur de *a*)
15. En cours, on a vu comment à l'aide de pointeurs définir une fonction `echange` qui échange les valeurs de deux variables entières. Pour échanger les valeurs des variables *a* et *b* on doit écrire :
- `echange(a, b) ;`
  - `echange(*a, *b) ;`
  - `echange(&a, &b) ;`
  - `echange(a++, b++) ;`

16. L'adresse d'une variable c'est :
- a. ce vers quoi pointe la variable
  - b. le contenu de la variable
  - c. le numéro de la case mémoire où le contenu de la variable est stocké
  - d. l'adresse d'un pointeur sur la variable
17. `if (a%2 == 0) printf("bonjour");`
- a. Affiche bonjour quand  $a$  est un entier pair
  - b. N'affiche rien (quelque soit la valeur de  $a$ )
  - c. Déclenche le message d'erreur `invalid lvalue in assignment`
  - d. Affiche bonjour quand  $a$  est un entier impair
18. On suppose que `a` a été déclarée par `int a`. L'expression `*a` a pour valeur :
- a. n'a pas de sens
  - b. l'adresse de la variable `a`
  - c. ce vers quoi pointe le pointeur `a`
  - d. la valeur de `a` tout simplement
19. `if (a<5) printf("Bonjour"); a=a+1;`
- a. N'affiche pas bonjour et n'augmente pas la valeur de  $a$  quelque soit  $a$
  - b. Affiche bonjour quelque soit  $a$
  - c. Augmente la valeur de  $a$  quelque soit  $a$
  - d. Affiche bonjour et augmente la valeur de  $a$  quelque soit  $a$
20. Après `char c; c='a'; c=c+1;`
- a. Un message d'erreur s'affiche
  - b. `c` vaut `'a'`
  - c. `c` vaut `'b'`
  - d. `c` vaut `'A'`