

---

## Partiel d'Éléments d'Informatique

---

**Durée :** 3 heures.

**Documents autorisés :** Aucun.

**Recommandations :** Un barème vous est donné à titre indicatif afin de vous permettre de gérer votre temps. La notation prendra en compte à la fois la syntaxe et la sémantique de vos programmes, c'est-à-dire qu'ils doivent compiler correctement. Une fois votre programme écrit, il est fortement recommandé de le faire tourner à la main sur un exemple pour s'assurer de sa correction.

### 1 Multiplication par additions successives (*5 points*)

Nous voulons écrire une fonction `mult_par_add` qui prend comme paramètres deux entiers positifs ou nuls `a` et `b`, et retourne la multiplication de `a` par `b`. Cette fonction doit calculer la multiplication **par additions successives, sans utiliser l'opérateur de multiplication du C**. Le programme principal est déjà écrit pour pouvoir tester la fonction :

```
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

int main()
{
    int a; /* premier facteur de la multiplication */
    int b; /* deuxieme facteur de la multiplication */

    printf("Entrez les deux facteurs a et b : ");
    scanf("%d",&a);
    scanf("%d",&b);

    while((a < 0) || (b < 0)) /* pas les deux positifs ou nuls */
    {
        printf("Erreur. Entrez les deux facteurs a et b : ");
        scanf("%d",&a);
        scanf("%d",&b);
    }

    printf("la multiplication de a par b est : %d\n",mult_par_add(a,b));
}
```

```

    return EXIT_SUCCESS;
}

```

1. Expliquer pourquoi le programme ci-dessus est incomplet et ce qu'il manque pour que la compilation réussisse, et puis pour que l'édition de liens réussisse. **Les réponses doivent être succinctement expliquées.**
2. Faire l'ajout minimal dans le programme pour que la compilation réussisse.
3. Écrire la fonction `mult_par_add`.

Un exemple de sortie du programme est :

```

Entrez les deux facteurs a et b : 5 -1
Erreur. Entrez les deux facteurs a et b : 5 4
La multiplication de a par b est : 20

```

### Correction.

- (2) 1 pt pour explication vague compil + 1 pt idem pour EdL
- 1 pour proto correct
- 2 pour code :
  - pas de boucle `-> 0`
  - 1 boucle correcte avec bonne borne (for ou while quoique c du for)
  - 0,5 return avec expression de bon type
  - 0,5 calcul expression correcte

## 2 Résolution de problèmes (*6 points*)

Il est demandé de résoudre les deux problèmes suivants sans définir de fonctions utilisateurs. L'ensemble du code sera à écrire dans la fonction principale `main`.

### 2.1 Calcul du produit des éléments d'un tableau (*2,5 points*)

Écrire le programme qui demande à l'utilisateur d'entrer les valeurs d'un tableau d'entiers de taille `N` (une constante symbolique), et qui calcule et affiche la valeur du produit des entiers du tableau. Deux exemples d'exécution sont les suivants (pour `N` valant 4) :

```

Saisissez 4 entiers : 2 -3 6 -10
Le produit des éléments du tableau vaut 360.

```

```

Saisissez 4 entiers : 3 -3 4 0
Le produit des éléments du tableau vaut 0.

```

tout ou rien : - 0.5pts : constante symbolique - 0.5 : squelette (programme vide) - 0.75 initialisation - 0.75 produit (incluant `init produit = 1`)

### Correction.

```

#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf, scanf */

```

```

/* déclaration constantes et types utilisateurs */
#define TAILLE 4 /* taille du tableau utilisateur */

/* déclaration de fonctions utilisateurs */

int main()
{
    int tab[TAILLE]; /* tableau à initialiser par l'utilisateur */
    int produit = 1; /* élément neutre de la multiplication */
    int i; /* var. de boucle */

    /* demande saisie de TAILLE entiers*/
    printf("Saisissez %d entiers : ",TAILLE);

    /* saisie des elts du tableau (TAILLE entiers) */
    for(i = 0;i < TAILLE;i = i + 1) /* chaque case du tableau */
    {
        /* saisie valeur */
        scanf("%d",&tab[i]);
    }
    /* i >= TAILLE */

    /* calcul produit */
    for(i = 0;i < TAILLE;i = i + 1) /* chaque case */
    {
        /* multiplie le produit partiel par la valeur de la case */
        produit = produit * tab[i];
    }
    /* i >= TAILLE */

    /* produit contient le produit des éléments du tableau */
    printf("Le produit des éléments du tableau vaut %d.\n",produit);

    return EXIT_SUCCESS;
}

/* définition de fonctions utilisateurs */

```

## 2.2 Une seconde plus tôt, il était exactement... (3,5 points)

Écrire un programme qui demande à l'utilisateur de saisir l'heure sous la forme de 3 entiers (3 variables pour les heures, h, les minutes, m et les secondes, s) et qui affiche l'heure qu'il était 1 seconde plus tôt. Il faudra envisager tous les cas possibles pour le changement d'heure. Deux exemples de sorties différentes sont :

```

Introduire l'heure puis les minutes puis les secondes : 23 12 0
Une seconde plus tôt, il était exactement : 23h11m59s

```

```

Introduire l'heure puis les minutes puis les secondes : 0 0 0
Une seconde plus tôt, il était exactement : 23h59m59s

```

## Correction.

(0.5 bonus)

une seconde plus TARD vu en TD. si ça -> 0

- 1 : saisie heure
- 1 : seconde OK
- 1 : minute OK
- 1 : heure OK

```
#include <stdlib.h> /* EXIT_SUCCESS */
```

```
#include <stdio.h> /* printf, scanf */
```

```
/* déclaration constantes et types utilisateurs */
```

```
/* déclaration de fonctions utilisateurs */
```

```
/* fonction principale */
```

```
int main()
```

```
{
```

```
    /* soit l'heure représentée par 3 entiers */
```

```
    int h; /* heures */
```

```
    int m; /* minutes */
```

```
    int s; /* secondes */
```

```
    /* saisie de l'heure */
```

```
    printf("Introduire l'heure puis les minutes puis les secondes : ");
```

```
    scanf("%d",&h);
```

```
    scanf("%d",&m);
```

```
    scanf("%d",&s);
```

```
    /* calcul de la nouvelle heure */
```

```
    s = s - 1; /* une seconde plus tôt */
```

```
    if(s == -1) /* tour du cadran à l'envers des secondes */
```

```
    {
```

```
        /* remise à 59 */
```

```
        s = 59;
```

```
        /* une minute de moins */
```

```
        m = m - 1;
```

```
        if(m == -1) /* tour du cadran à l'envers des minutes */
```

```
        {
```

```
            /* remise à 59 */
```

```
            m = 59;
```

```
            /* une heure de moins */
```

```
            h = h - 1;
```

```
            if(h == -1) /* tour du cadran à l'envers des heures */
```

```
            {
```

```

        /* remise à 23 */
        h = 23;
    }
}
/* h,m,s contiennent l'heure une seconde plus tôt */

/* affichage de l'heure */
printf("Une seconde plus tôt, il était exactement : %dh%dm%ds\n",h,m,s);

return EXIT_SUCCESS;
}

/* définition de fonctions utilisateurs */

```

### 3 Trace d'un programme avec fonctions (*4 points*)

Simulez l'exécution du programme suivant, en réalisant sa **trace**, comme cela a été vu en TD et en cours. **Rappel** : L'opérateur modulo du C % calcule le reste de la division entière : a % b vaut le reste de la division entière de a par b.

```

1  #include <stdlib.h> /* EXIT_SUCCESS */
2  #include <stdio.h> /* printf, scanf */
3
4  /* declarations constantes et types utilisateurs */
5
6  /* declarations de fonctions utilisateurs */
7  int mystere(int a, int b);
8
9  int main()
10 {
11     int x = 12;
12     int y = 4;
13     int res;
14
15     res = mystere(x, y);
16     printf("mystere(%d, %d) = %d\n", x, y, res);
17
18     return EXIT_SUCCESS;
19 }
20
21 /* definitions de fonctions utilisateurs */
22 int mystere(int a, int b)
23 {
24     if (b == 0)
25     {
26         return a;
27     }
28     return mystere(b, a % b);

```

## 4 Écriture de fonctions ( $5 \times 1$ point)

1. Écrire la fonction `factorielle` qui prend en entrée un entier positif ou nul  $n$  et qui renvoie sa factorielle  $n!$ , avec la convention  $0! = 1$ .
2. Écrire la fonction `combinaison` qui calcule le nombre de combinaisons possibles quand on choisit  $k$  objets parmi  $n$  objets discernables et que l'ordre dans lequel les objets sont placés (ou énumérés) n'a pas d'importance : elle prend en entrée deux entiers  $k$  et  $n$  et renvoie  $\frac{n!}{k!(n-k)!}$ . On considérera la fonction `factorielle` comme disponible.
3. Écrire la procédure `affiche_menu` qui n'a pas d'entrée et affiche le menu suivant :

Votre choix ?

1) choix 1

2) choix 2

3) choix 3

4. Écrire la procédure `rectangle_d_etoiles` qui prend en entrée la longueur et la largeur en nombre d'étoiles (caractère `'*'`) du rectangle, et qui affiche le rectangle d'étoiles.
5. Écrire la fonction qui prend en entrée un entier  $n$  et qui renvoie :

$$\sum_{i=0}^{i=n-1} \left( \sum_{j=0}^{j=i} i * j \right)$$

**Correction.** 0.5 points si proto juste 0.5 si corps entierement juste (tolère 1 faute syntaxique anodine, genre oubli ;)