Université Paris-Nord — L1
Institut Galilée — Année 2009-2010
Licence 1 - 1er semestre

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 succintement 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
```

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.
```

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
```

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);
29  }
```

4 Écriture de fonctions (*5 × 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 :

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