Licence Professionnelle EON, contrôle des connaissances : Programmation en C
— Corrigé —

Luca Saiu — novembre 2018

Aucun document autorisé. Toute communication entre étudiants est interdite. Téléphones portable et ordinateurs éteintes et rangés.

Le barème est donné à titre indicatif. ***Il a une pénalité pour toute réponse incorrecte*** *: ne rien écrire est préférable à une erreur. Toute réponse ambiguë sera considérée comme incorrecte.*

Les réponses sont à donner directement sur le papier de l'énoncé.

Durée : 90 minutes. La partie B est au verso.

Les programmes ou fonctions suivants, même quand ils sont corrects, peuvent être mal décalés (« indentées ») ; du code mal décalé **n'est pas** considéré comme erroné ou incorrect. Faites attention aux point-virgules ! Du code qui ne termine jamais, également, **n'est pas** considéré comme erroné ou incorrect.
Il y a une différence entre une fonction renvoyant un résultat est une commande affichant du texte sur le terminal ; il y a une différence entre une fonction acceptant un paramètre et une valeur lue à partir du terminal. Suivez **exactement** les consignes de l'énoncé.

Nom (lisible) : _____

Prénom (lisible) : _____

— Partie A : compréhension —

(chaque question a une seule réponse correcte)

Question A.1. *(2 points)* La phrase du langage C

```
return (a + 3);
```

a) est une expression
b) est une commande
c) est un bloc
d) est un prototype (ou déclaration) de fonction
e) est une définition de fonction
f) est un appel de fonction
g) est mal écrite

`return` *`expression`* `;` est syntaxiquement une commande.

Question A.2. *(2 points)* Le programme suivant . . .

```
1  #include <stdio.h>
2
3  int
4  main (void)
5  {
6    int i;
7    for (i = 0; i < 10; i ++)
8    print ("foo\n");
9    print ("foo\n");
10   return 0;
11 }
```

a) affiche `foo` 9 fois
b) affiche `foo` 10 fois
c) affiche `foo` 11 fois
d) affiche `foo` 12 fois
e) affiche `foo` 20 fois
f) affiche `foo` 100 fois
g) est mal écrit
h) ne termine jamais

Le code est (intentionnellement) mal décalé, ou mal « indenté ». Juste la ligne 8 fait partie du corps de la boucle ; la ligne 9 est exécutée une seule fois, après la boucle. Donc le programme affiche `foo` 11 fois : 10 fois dans la boucle, puis une autre fois dehors.

Question A.3. *(2 points)* Le programme suivant . . .

```
1  #include <stdio.h>
2
3  int
4  main (void)
5  {
6    int i;
7    int n = 1;
8    for (i = 0; i < 4; i ++)
9      n = n + i;
10   printf ("%i\n", n);
11   return 0;
12 }
```

a) affiche `3`
b) affiche `4`
c) affiche `5`
d) affiche `6`
e) affiche `7`
f) affiche `8`
g) affiche `9`
h) affiche `10`
i) est mal écrit
j) ne termine jamais

Le programme calcule 1 (la valeur initiale de `n`) + 0 + 1 + 2 + 3, donc à la fin `n` sera à 7.

Question A.4. *(3 points)* La fonction `foo` suivante . . .

```
1  void
2  foo (int n)
3  {
4    if (n != 0)
5      n = 0;
6    while (n < 3)
7      n = 0;
8    return n;
9  }
```

a) renvoie la valeur initiale de `n`
b) renvoie toujours `0`
c) renvoie toujours `2`
d) renvoie toujours `3`
e) renvoie toujours `4`
f) est mal écrit
g) ne termine jamais

`n`, à l'entrée de la fonction, sera soit 0 soit une valeur différente de 0. En tout cas, après la conditionnelle aux lignes 4 et 5, `n` est mise à 0, si elle n'était pas déjà 0. Donc à l'entrée de la boucle on a que `n == 0`, et certainement `n < 3`. Le corps de la boucle remets `n` à 0, et donc `n` ne change plus de valeur, en restant toujours mineure de 3. La boucle est infinie, et on n'arrive jamais à la ligne 8.

Question A.5. *(3 points)* La fonction `foo` [erreur dans l'énoncé : la fonction est, bien sûr, `quux`] suivante . . .

```
1  int
2  quux (bool frob, int x)
3  {
4    while (frob)
5      x = 1;
6    return 7;
7  }
```

a) renvoie toujours `true`
b) renvoie toujours `false`
c) renvoie toujours `7`
d) renvoie toujours la valeur de `x`
e) peut renvoyer `7` ou boucler infiniment, selon la valeur de `frob`
f) peut renvoyer `7` ou renvoyer `true`, selon la valeur de `frob`
g) peut renvoyer `7` ou renvoyer `false`, selon la valeur de `frob`
h) est mal écrit
i) ne termine jamais

Si `frob` est vraie, la boucle est infinie : exécuter la ligne 5 ne changera jamais la valeur de `frob`, donc on reste toujours dans la boucle. Si par contre `frob` est fausse on ne rentre pas dans la boucle, et on renvoie simplement 7.

— Partie B : programmation —

(ce n'est pas nécessaire de répondre à toute question de cette partie pour avoir la note maximale, si le reste est correct)

Question B.1. *(7 points)* Écrivez la définition d'une fonction `m` qui, étant donné un paramètre entier `n`, affiche une *table de multiplication* de 1 à `n` compris, sans renvoyer aucun résultat, dans un format similaire à ce qui suit. Par exemple l'appel `m (3)` affichera :

```
1 fois 1 = 1
1 fois 2 = 2
1 fois 3 = 3
2 fois 1 = 2
2 fois 2 = 4
2 fois 3 = 6
3 fois 1 = 3
3 fois 2 = 6
3 fois 3 = 9
```

Voilà une solution possible :

```
1  void
2  m (int n)
3  {
4    int i;
5    int j;
6    for (i = 1; i <= n; i ++)
7      for (j = 1; j <= n; j ++)
8        printf ("%i fois %i = %i\n", i, j, i * j);
9  }
```

Ces qui ne sont pas à l'aise avec `printf` peuvent aussi remplacer la ligne 8 ici en haut par un bloc comme

```
1        {
2          printf ("%i", i);
3          printf (" fois ");
4          printf ("%i", j);
5          printf (" = ");
6          printf ("%i", i * j);
7          printf ("\n");
8        }
```

, ce qui bien sûr est également correct.

Question B.2. *(7 points)* Écrivez la définition d'une fonction `s` qui, étant donné deux paramètres entiers `a` et `b`, renvoie la somme des nombres naturels entre 1 et `a` (compris) qui sont des multiples de `b`. Par exemple l'appel `s (10, 3)` renverra `18`, car $3 + 6 + 9 = 18$ et les seuls multiples de 3 dans $\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ sont 3, 6 et 9.

Voilà une solution possible :

```
1  int
2  s (int a, int b)
3  {
4    int res = 0;
5    int i;
6    for (i = 1; i <= a; i ++)
7      if (i % b == 0)
8        res += i;
9    return res;
10 }
```

Une autre solution, peut-être encore plus simple :

```
1  int
2  s (int a, int b)
3  {
4    int res = 0;
5    int i;
6    for (i = b; i <= a; i += b)
7      res += i;
8    return res;
9  }
```