

---

## Éléments d'informatique : partiel de fin de semestre

---

**Durée :** 3 heures.

**Documents autorisés :** Aucun.


**Recommandations :** Un barème vous est donné à titre indicatif, afin de vous permettre de gérer votre temps. Ne dépassez pas le temps indiqué pour chaque question. Par contre, vous pouvez tout à fait répondre beaucoup plus rapidement. 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 recommandé de le faire tourner à la main sur un exemple pour s'assurer de sa correction.

### 1 For ou while ? (7 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`.

#### Tableau (1,5 points)

**Question A.** Soit un tableau d'entiers non initialisé et dont la taille sera fixée à l'aide d'une constante symbolique `N`. Écrire un programme qui initialise chaque case  $i$  du tableau à la valeur  $i^2 + 1$  ( $i$  est l'indice de la case).

 1,5 pt  
13 min

#### Pour une poignée de brouzoufs (5,5 points)

Pour le bon fonctionnement des machines à café de notre base lunaire, nous devons programmer une machine à rendre la monnaie. L'unité monétaire lunaire est le brouzouf (nom fictif). Il y a des pièces de 10, 20 ou 50 brouzoufs, et les montants acceptés par la machine sont des multiples de 10.

Un fois votre programme construit, l'utilisateur pourra saisir un montant et verra s'afficher toutes les manières dont nous pouvons lui faire la monnaie sur ce montant. Dans l'exemple suivant, l'utilisateur saisit 60 :


Entrer un montant : 60

Pour rendre 60 brouzoufs, je peux donner :

```
* 0 en pieces de 50 + 0 en pieces de 20 + 60 en pieces de 10
* 0 en pieces de 50 + 20 en pieces de 20 + 40 en pieces de 10
* 0 en pieces de 50 + 40 en pieces de 20 + 20 en pieces de 10
* 0 en pieces de 50 + 60 en pieces de 20 + 0 en pieces de 10
* 50 en pieces de 50 + 0 en pieces de 20 + 10 en pieces de 10
```

Il y a 5 facons de rendre la monnaie.


**Question B.** Commencer par écrire un programme qui demande à l'utilisateur de saisir un montant  $x$ , sans vérifier la validité de la saisie (nous supposons que l'utilisateur saisit toujours un multiple de 10), puis affiche à l'utilisateur tous les multiples  $m$  de 50 inférieurs ou égaux à ce montant  $x$ .

 1.5 pt  
13 min


Ceci nous indique toutes les possibilités de rendre une partie de la monnaie en pièces de 50, et à chaque fois, il reste à rendre la monnaie sur un montant de  $x - m$ . Pour rendre le reste de la monnaie, il suffit de déterminer tous les multiples de 20 inférieurs au montant  $x - m$ , puis pour chaque possibilité de compléter avec des pièces de 10.

Pour les deux questions suivantes vous pouvez n'indiquer que les modifications que vous apportez au programme précédent.

**Question C.** Écrire le programme complet, mais sans vérifier la validité de la saisie. Votre programme devra afficher toutes les manières possible de rendre la monnaie et le nombre de façons de le faire (comme dans l'exemple).


 3 pt  
27 min

**Question D.** Modifier la saisie de manière à ce que tant que le montant n'est pas un multiple de 10, l'utilisateur doive saisir de nouveau ce nombre.


 1 pt  
9 min

## 2 Trace d'un programme avec fonctions (5 points)


**Question E.** Simulez l'exécution du programme figure 1, en réalisant sa **trace**, comme cela a été vu en TD et en cours.

 4 pt  
36 min

**Question F.** Réécrire les lignes 30 à 35 avec un **for** au lieu du **while** sans changer la sémantique du programme (le code machine généré).


 0,5 pt  
4 min

**Question G.** Les deux fonctions `C foo` et `bar` calculent la même fonction mathématique, mais une seule est récursive. Laquelle ? Rappeler brièvement ce qu'est une fonction récursive.


 0,5 pt  
4 min

## 3 Points du plan (4,5 points)


**Question H.** Déclarer un type utilisateur `point_s` pour représenter les points du plan réel en coordonnées cartésiennes (les couples  $(x, y)$  avec  $x \in \mathbb{R}$  et  $y \in \mathbb{R}$ ).

 1 pt  
9 min


**Question I.** Déclarer et définir une fonction `calculer_milieu` prenant en paramètres deux points du plan  $a$  et  $b$  et retournant les coordonnées du point situé au milieu du segment  $[a, b]$ . Rappel : si  $(x_a, y_a)$  sont les coordonnées de  $a$  et  $(x_b, y_b)$  les coordonnées de  $b$ , alors le milieu de  $[a, b]$  a pour coordonnées  $(\frac{x_a+x_b}{2}, \frac{y_a+y_b}{2})$ .

 1.5 pt  
13 min

**Question J.** Déclarer et définir une fonction `calculer_distance` prenant en paramètres deux points du plan  $a$  et  $b$  et retournant la distance qui sépare  $a$  et  $b$ . Vous pourrez utiliser les fonctions `sqrt(double x)` et `pow(double base, double exposant)` de la bibliothèque `math` pour effectuer le calcul.

 1.5 pt  
13 min

**Question K.** Quelle instruction préprocesseur vous permet de vous assurer que les fonctions de la bibliothèque `math` sont bien déclarées ?

 0.5 pt  
4 min

```

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 foo(int n);
8  int bar(int n);
9
10 /* fonction principale */
11 int main()
12 {
13     int x = 1;
14     int res;
15
16     res = foo(x);
17     printf("foo(%d) = %d\n", x, res);
18
19     res = bar(x);
20     printf("bar(%d) = %d\n", x, res);
21
22     return EXIT_SUCCESS;
23 }
24
25 /* definitions de fonctions utilisateurs */
26 int foo(int n)
27 {
28     int i;
29     int res = 5;
30     i = 0;
31     while(i < n)
32     {
33         res = 3 * res + 2;
34         i = i + 1;
35     }
36     return res;
37 }
38
39 int bar(int n)
40 {
41     if (n == 0)
42     {
43         return 5;
44     }
45     return 3 * bar(n - 1) + 2;
46 }

```

FIGURE 1 – Programme pour la trace

## 4 Énumération des nombres premiers (3,5 points)

Pippo dispose d'une fonction `int est_premier(int x)` qui renvoie *vrai* si  $x$  est premier et *faux* sinon. Rappel : l'entier 1 n'est pas considéré comme premier.

Pippo souhaite disposer d'une énumération des nombres premiers par ordre croissant comme dans le tableau suivant.


numéro	1	2	3	4	5	6	7	8	9	...
nombre premier	2	3	5	7	11	13	17	19	23	...

Mais Pippo ne veut pas mémoriser de tableau dans son programme. Il veut disposer de cette énumération sous la forme d'une fonction dont la déclaration sera :


```
int numero_vers_premier(int n);
```

Cette fonction prend en paramètre un numéro  $n$  (un entier positif non nul) et retourne le  $n$ -ième nombre premier. Par exemple, l'appel `numero_vers_premier(9)` lui retournera le neuvième nombre premier (qui est 23).


**Question L.** Définir la fonction `numero_vers_premier` (vous pouvez faire appel à la fonction `est_premier`).

 2.5 pt  
22 min

**Question M.** Définir la fonction `est_premier` (comme en cours et en TD).

 1 pt  
9 min

**Question bonus (plus difficile).** Déclarer et définir la fonction réciproque de la précédente, `premier_vers_numero` (qui prend en entrée un nombre premier et renvoie son numéro d'ordre).

 2 pt  
18 min