
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

Correction.

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

/* declarations constantes et types utilisateurs */
#define N 3

/* fonction principale */
int main()
{
    int t[N];
    int i; /* variable de boucle */

    for (i = 0; i < N; i = i + 1) /* pour chaque case */
    {
        /* initialiser la case */
        t[i] = i * i + 1;
    }

    return EXIT_SUCCESS;
}
```

Barème. *Maximum :* 1, 5 pt.

- (a) +0, 5 pt pour la déclaration de la constante symbolique
 - (b) +0, 25 pt si déclarations des deux variables (*i* et *t[N]*)
 - (c) +0, 5 pt si le tableau est parcouru par une boucle *for* (avec ou sans la constante symbolique pour la dimension).
 - (d) +0, 25 pt affectation correcte des valeurs des cases du tableau.
- * Pas de pénalité pour mauvais *include* ou *return*.

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

Correction.

```
/* Déclaration de fonctionnalités supplémentaires */
#include <stdlib.h> /* EXIT_SUCCESS */
#include <stdio.h> /* printf */

/* Déclaration constantes et types utilisateurs */

/* Déclaration de fonctions utilisateurs */

/* Fonction principale */
int main()
{
    /* Déclaration et initialisation variables */
    int i;
    int n = 0;

    printf("Entrer un montant : ");
    scanf("%d", &n);

    printf("multiples de 50 inferieurs a %d brouzoufs :\n", n);
```

```

    for (i = 0; i <= n; i = i + 50) /* i en pieces de 50 */
    {
        printf("%d\n", i);
    }

    return EXIT_SUCCESS;
}

/* definitions des fonctions utilisateurs */

```


Barème. Maximum : 1,75 pt. (hausse du barème). Il fallait faire deux choses : la saisie (0,5 pt) et surtout la boucle. L’affichage aurait du être celui des multiples de 50, mais on compte presque tous les points si c’est le nombre de pièces de 50 qui est affiché.

- (a) +0,5 pt pour la saisie.
- (b) 0,5 pt si il y a exactement une boucle.
- (c) +0,5 pt si la boucle parcourt les multiples successifs de 50 ou si elle compte le nombre de pièces de 50.
- (d) +0,25 pt si il y a affichage des multiples.
- (e) −0,25 pt si l’esperluette est manquante dans la saisie, si une variable est non déclaré, si l’une des bornes de la boucle est mauvaise (d’un pas).

Ceci nous indique toute 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

Correction.

```

1  /* Déclaration de fonctionnalités supplémentaires */
2  #include <stdlib.h> /* EXIT_SUCCESS */
3  #include <stdio.h> /* printf */
4
5  /* Déclaration constantes et types utilisateurs */
6
7  /* Déclaration de fonctions utilisateurs */
8
9  /* Fonction principale */
10 int main()
11 {
12     /* Déclaration et initialisation variables */
13     int i;
14     int j;
15     int compt = 0;
16     int n = 0;
17
18     printf("Entrer un montant : ");
19     scanf("%d", &n);

```

```


20
21     printf("Pour rendre %d brouzoufs, je peux donner :\n", n);
22     for (i = 0; i <= n; i = i + 50) /* i pieces de 50 */
23     {
24         for (j = 0; j <= n - i; j = j + 20) /* j pieces de 20 */
25         {
26             compt =compt + 1;
27             printf("* %d en pieces de 50 + %d en pieces de 20 + %d en %pieces de 10\n",
28                   i, j, n - i - j);
29         }
30     }
31
32     printf("Il y a %d facon de rendre la monnaie\n", compt);
33
34     return EXIT_SUCCESS;
35 }
36
37 /* definitions des fonctions utilisateurs */

```

Barème. *Il fallait surtout ne pas se tromper en imbricant les deux boucles.*

- (a) 1 pt *si il y a exactement deux boucles imbriquées.*
- (b) +0,5 pt *si la seconde boucle compte bien les multiples de 20 jusqu'au montant sur le lequel il reste à rendre la monnaie.*
- (c) +0,5 pt *si la monnaie est complétée correctement par des pièces de 10.*
- (d) +0,5 pt *Si l'affichage est à l'intérieur de la boucle comprend les 3 valeurs.*
- (e) +0,5 pt *si il y a un compteur correctement initialisé et incrémenté à chaque tour de la boucle interne.*

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

 1 pt
9 min

Correction. On se donne une variable booléenne qui dénotera le fait que la saisie est correcte ou incorrecte. Tant que cette variable est à faux on redemande la saisie et on vérifie si elle est correcte.

On remplace les lignes 17 à 19 par :

```

int saisie = FALSE;

while (!saisie)
{
    printf("Entrer un montant : ");
    scanf("%d", &n);
    if (0 = n % 10)
    {
        saisie = TRUE;
    }
    else
    {
        printf("Erreur : %d n'est pas multiple de 10\n", n);
    }
}

```

Et on ajoute ligne 6 les incontournables :

main()			
ligne	x	res	Affichage
initialisation	1	?	
16			
16		17	
17			foo(1) = 17 \n
19			
19		17	
20			bar(1) = 17 \n
22	renvoie EXIT_SUCCESS		

foo(1)			
ligne	n	i	res
initialisation	1	?	5
30		0	
33			17
34		1	
36	renvoie 17		

bar(1)	
ligne	n
initialisation	1
45	
45	renvoie 17

bar(0)	
ligne	n
initialisation	0
43	renvoie 5

TABLE 1 – Trace du programme de l'exercice 2.

```
#define TRUE 1
#define FALSE 0
```


Barème. *Maximum :* 1 pt.

* Ici, on ne sanctionne pas l'absence de l'esperluette dans le scanf.

- (a) +1 pt On donne 1 point pour n'importe quelle solution opérationnelle (tant qu'elle n'est pas trop tirée par les cheveux – pas de double boucle!). L'affichage n'a pas à être parfait (la partie qui est ici dans le else peut être absente).
- (b) −0,5 pt Si la construction du test « n est multiple de 10 » est fausse on enlève un demi-point. Il n'est pas nécessaire de passer par une variable booléenne, mais si c'est le cas elle doit être correctement déclarée et initialisée. On ne sanctionne pas l'absence des déclarations des constantes (symboliques) booléennes,
- (c) +0,25 pt mais à défaut de trouver une solution juste on peut compter un quart de point pour ces déclarations de variables booléennes.

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

Correction. Table 1 page 5.

```

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

Barème. Maximum 4 pt. Si des erreurs, maximum 3,75 pt.

- (a) +1 pt deux premières lignes de la trace du main sont correctes (identification des variables et leurs initialisations).
- * -0,5 pt par variable manquante ou en trop
 - * -0,5 pt par initialisation manquante ou en trop
- (b) +1,25 pt Pour l'appel à foo :
- * +0,25 pt pour foo(1) ligne 16
 - * +0,25 pt une colonne pour le paramètre formel n bien initialisé à 1
 - * +0,25 pt pour le déroulement correct de la boucle (un tour, $i = 1$)
 - * +0,25 pt pour le retour d'un entier (même si valeur fausse)
 - * +0,25 pt ligne 16 affectation en accord avec cette valeur
- (c) +1,25 pt pour l'appel récursif à bar :
- * +0,25 pt pour bar(1) ligne 19
 - * +0,5 pt pour le déclenchement du second appel ligne 45.
 - * +0,25 pt une colonne pour le paramètre formel n bien initialisé à 1 et à 0 dans le second appel.
 - * +0,25 pt pour le retour de la valeur et l'affectation ligne 19.
- (d) +0,25 pt pour au moins l'un des deux affichages (ligne 17 et ligne 20) sur valeur cohérente avec la case.

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


Correction.

```
30         for (i = 0; i < n; i = i + 1)
31         {
32             res = 3 * res + 2;
33         }
34
35
36
```

Barème. Maximum 0,5 pt.

- (a) 0,5 pt si la boucle est exactement écrite comme ci-dessus.
- (b) Zéro sinon.

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


Correction. Une fonction récursive est une fonction dont la définition fait appel à la fonction elle-même. La fonction bar est récursive car sa définition (lignes 39 à 46) fait appel à la fonction bar (ligne 45).

Barème. Maximum 0,5 pt.

- (a) 0,5 pt si le rappel de cours est correct et si bar est correctement identifié comme fonction récursive.
- (b) 0,25 pt si seul l'un des deux est correct.
- (c) Zéro sinon.

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

Correction. On déclare le type `point_s`, comme un struct :

```
struct point_s
{
    double x; /* abscisse */
    double y; /* ordonnee */
};
```

Barème. Maximum 1 pt (tout juste). On ne compte pas faux si le point virgule final est oublié. Sinon zéro.

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

Correction. Déclaration :

```
struct point_s calculer_milieu(struct point_s a, struct point_s b);
```

Définition :

```
struct point_s calculer_milieu(struct point_s a, struct point_s b)
{
    struct point_s milieu;

    milieu.x = (a.x + b.x) / 2.0;
    milieu.y = (a.y + b.y) / 2.0;

    return milieu;
}
```

Barème. Maximum 1,5 pt.

- (a) +0,5 pt déclaration correcte.
- (b) 1 pt définition correcte.
- (c) 0,5 pt autrement, si la définition retourne un point mais qui n'a pas la bonne valeur.

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

Correction. Déclaration :

```
double calculer_distance(struct point_s a, struct point_s b);
```

Pour définir la fonction il faut se souvenir que la distance sera

$$d(a, b) = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

Définition :

```
double calculer_distance(struct point_s a, struct point_s b)
{
    double d;

    d = sqrt(pow(x.a - x.b, 2.0) + pow(y.a - y.b, 2.0));

    return d;
}
```

Barème. Maximum 1,5 pt. Si incorrect maximum 1 pt.

- (a) +0,5 pt déclaration correcte.
- (b) 1 pt définition correcte (qui peut ne pas faire appel à `pow`).
- (c) 0,5 pt si la définition retourne un `double` mais qui n'a pas la bonne valeur (par exemple, erreur de définition mathématique de la distance).

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

Correction. L'instruction `#include <math>` est nécessaire, au début du programme et en début de ligne, pour charger les déclarations des fonctions de la bibliothèque mathématique.

Barème. 0,5 pt si mentionne l'instruction `include`, sinon zéro. un `-lm` est hors sujet.

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


Correction.

```
int numero_vers_premier(int n)
{
    int compt = 0; /* compteur */
    int p = 1; /*var de boucle */
    while (compt < n)
    {
        p = p + 1;
        if (est_premier(p))
        {
            compt = compt + 1; /*on compte un nouveau nb premier */
        }
    }
    /* le compteur a enregistre n nombre premiers, p est le dernier d'entre eux */
    return p;
}
```

Barème. Maximum : 2,5 pt. Si incorrect maximum 2 pt.

- * La difficulté était algorithmique, on note essentiellement la dessus. Il faut nécessairement une boucle et un compteur sinon zéro.
- (a) 1 pt. Compter 1 point si présence d'un compteur et de exactement une boucle (ormis d'éventuelles boucles pour réinventer le test de primalité au cas où la réponse ne fasse pas appel à `est_premier`).
- (b) +0,5 pt. Ajouter 0,5 points si la condition d'arrêt correspond à un compteur arrivant à n .
- (c) +0,5 pt Compter 0,5 points en plus si de plus l'incrément du compteur est conditionnée à un test de primalité, (c'est à dire `if` dans la boucle avec un appel à `est_premier` ou un code ressemblant à un test de primalité), même si il y a des conditions supplémentaires inutiles.

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

 1 pt
9 min

Correction. Nous avons besoin des constantes booléennes. On place en début de programme :

```
#define FALSE 0
#define TRUE 1
```

On écrit la fonction comme ceci :

```
int est_premier(int n)
{
    int i;
```

```

for (i = 2; i < n; i = i + 1)
{
    if (n % i == 0)
    {
        return FALSE;
    }
}
return TRUE;
}

```

Barème. *C'est un rappel de cours et TD immédiat : C'est donc soit tout juste soit vraiment pas terrible. Maximum : 1 pt.*

Si la réponse est incorrecte :

- (a) +0,25 pt *on peut compter 0,25 point si les constantes booléennes sont définies et/ou utilisées*
- (b) +0,25 pt *s'il y a exactement une boucle.*

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

Correction.

```

int premier_vers_numero(int p)
{
    int compt = 0; /* compteur */
    int q = 1; /*var de boucle */
    while (q < p)
    {
        q = q + 1;
        if (est_premier(q))
        {
            compt = compt + 1; /*on compte un nouveau nb premier */
        }
    }
    /* le compteur a enregistre tous les nombre premiers jusqu'a p compris */
    return compt;
}

```

Barème. *Maximum : 2 pt.*

- (a) 0 pt *Ne pas compter de point s'il y a la moindre erreur algorithmique.*
- (b) 1 pt *On peut compter 1 point si l'algo est juste mais qu'il y a de grosses erreurs de programmation (manque une déclaration etc.).*