Examen de Programmation C

ING1 – GM

12 janvier 2022

— *Durée : 2 heures*

— *Vous devez rédiger votre copie à l'aide d'un **stylo à encre** exclusivement.*

— *Toutes vos affaires (sacs, vestes, trousse, etc.) doivent être placées à l'avant de la salle.*

— *Aucun document n'est autorisé.*

— *Aucune machine électronique ne doit se trouver sur vous ou à proximité, même éteinte.*

— *Aucun déplacement n'est autorisé.*

— ***Aucune question au professeur n'est autorisée**. Si vous pensez avoir détecté une erreur, continuez en expliquant les hypothèses que vous faites.*

— *Aucun échange, de quelque nature que ce soit, n'est possible.*

— *Le barème est donné à titre indicatif.*

Exercice 1 (4.5pts = 11*0.25 + 7*0.25)

1. Donner le résultat d'exécution du programme C suivant :

```c
#include <stdio.h>
int main(void){
    int r;
    int m;
    int n;
    int *p;
    int *q;
    m = 40;
    n = 10;
    if( m > n ) {
        r = (m%=3);
    } else {
        r = (n%=5);
    }
    printf("\n m = %d n = %d r = %d", m, n, r);
    m = 4;
    n = 5;
    p = &m;
    q = &n;
    *q = m + n;
```

```
21      q = p;
22      printf("\n m = %d n = %d *p = %d *q = %d", m, n, *p, *q);
23      m = 6;
24      n = 2;
25      p = &m;
26      q = p;
27      n = (*q) + 13;
28      m = 13 - (*p);
29      printf("\n m = %d n = %d *p = %d *q = %d", m, n, *p, *q);
30      return (0);
31  }
```

2. Soit *p* un pointeur qui *pointe* sur un élément d'un tableau `tab` d'entiers :

```
1  int tab[DIM] = {12, 23, 14, 45, 56, 67, 78, 89, 92};
2  int *p;
3  p = &tab[2];
```

Quelles valeurs ou adresses fournissent les expressions suivantes :

(a) *p+2
(b) *(p+2)
(c) tab + 3
(d) &tab[7] - &tab[1]
(e) &tab[7] - p
(f) p+(*p-10)
(g) *(p+*(p+6)-tab[7])

Exercice 2 (3 pts = 0.5 + 1.25 + 0.75 + 0.5)

```
1  #include <stdio.h>
2  #define DIM 13
3  typedef int TElement;
4  //Pre-condition: b > a
5  int toto (int n, TElement a, TElement b, TElement *tab){
6      int i;
7      TElement s;
8      s = 0;
9      for (i=0; i<n; i++) {
10         if (tab[i]>=a && tab[i]<=b) {
11             s++;
12         }
13     }
14     return (s);
15 }
```

```
16 void main (void) {
17     TElement tab[DIM]={8,5,4,9,1,6,3,5,7,0};
18     int n;
19     int a;
20     int b;
21     n = 10;
22     a = 4;
23     b = 7;
24     printf("\n Appel fct toto = %d",toto(n, a, b tab));
25 }
```

1. Donner le résultat d'exécution du programme ci-dessus.
2. En utilisant le formalisme pointeur (aucun indice entier ne doit être utilisé pour manipuler le tableau) ; réécrire la fonction `toto`.
3. Réécrire la fonction `toto` sous forme de procédure.
4. Modifier le programme principal en faisant appel à la procédure `toto`.

Exercice 3 (3.5 pts = 2 + 1.5)

1. Donner le résultat d'exécution du programme C suivant :

```
1  #include <stdio.h>
2  int tata(int a, int b) {
3      return (a*a + b*b);
4  }
5  int toto(int * a, int b) {
6      *a = tata(*a, b);
7      return (2*(*a+b));
8  }
9  void tutu(int a, int * b) {
10     a = *b;
11     *b = 3*a;
12 }
13 int Un(int n){
14     return (n-5);
15 }
16 int Vn(int n ){
17     return (-2*n+10);
18 }
```

```c
int rePremierEntier(int n){
    if(Un(n) >= Vn(n)) {
        return (n);
    } else {
        return (rePremierEntier(n+1));
    }
}
void main(void) {
    int i;
    int j;
    int k;
    i = 1;
    j = 2;
    k = 10;
    tutu(i, &j);
    printf("\n i = %d j = %d", i, j);
    i = 3;
    j = 3;
    k = toto(&i, j);
    printf("\n i = %d j = %d k = %d", i, j, k);
    printf("\n %d ", rePremierEntier(0));
}
```

2. Le programme principal (fonction main) ci-dessous comporte des erreurs.
   Indiquer les erreurs et justifier pourquoi?

```c
#include <stdio.h>
void tata(double d,double *x, double *y){
    *x += d;
    *y + =d;
}
int toto(int n){
    int i;
    int s;
    s = 0;
    for(i=0; i<n; i++) {
        s += i;
    }
    return (s);
```

```
14 }
15 void main(void){
16     double m;
17     double x;
18     double y;
19     int *p;
20     int *q;
21     m = 4;
22     x = 1;
23     y = 2;
24     x = tata(4, 5, 6);
25     tata(2, 2*x, 2*y);
26     p = toto(&m);
27     *q = toto(x+y);
28 }
```

Exercice 4 (2 pts = 1 + 1)

```
1  #include <stdio.h>
2  #define DIM 13
3  typedef float TElement;
4  int toto(int n, TElement *tab) {
5      TElement *p;
6      TElement *M;
7      p = tab;
8      M = tab+n-1;
9      while( (p<M) && (*p <= *(p+1))) {
10         p++;
11     }
12     return (p == M);
13 }
14 void main(void) {
15     TElement tab[DIM]={2.2, 4.4, 6, 0.9, 5, 3.3};
16     int n;
17     n = 6;
18     printf("\n toto = %d", toto(n, tab));
19 }
```

1. **Illustrer à chaque itération**, l'exécution de la fonction `toto` écrite en formalisme pointeur.
2. En utilisant le formalisme tableau, réécrire la fonction `toto`.

Problème (7 pts)

Le but de ce problème est la gestion des ensembles en utilisant une représentation contiguë.
Un **ensemble** est constitué d'un nombre fini **sans répétitions** de ses éléments. **On suppose que les éléments de l'ensemble ne sont pas triés**.
On représente le type ensemble par la structure dynamique suivante :

```
1 typedef float TElement;   // le type des elements de l'ensemble
2 typedef struct {
3     int NbElemMax;   // nbre d'elements max qu'un ensemble peut contenir.
4     int NbElem;      // nombre d'elements existant dans l'ensemble.
5     TElement *tab;   // stockage des elements de l'ensemble.
6 } Ensemble;
```

*Dans ce qui suit, on utilise le mot **algorithme** pour désigner soit une **fonction** soit une **procédure**.*
Les algorithmes ci-dessous doivent être écrits en C.

1. Écrire les codes des fonctions suivantes qui sont liées à la structure de donnée `Ensemble` :
   - `int nbreEltMaxE(Ensemble e);` *//Retourne le nombre maximum d'éléments présent dans un ensemble donné.*
   - `int nbEltsE(Ensemble e);` *//Retourne le nombre d'éléments présent dans un ensemble donné.*
   - `TElement iEmeEltE(int i, Ensemble e);` *//Retourne le iéme élément d'un ensemble non vide donné et on suppose que 0 < i < nbEltsE(e)*
2. Écrire un algorithme nommé **`allocMemTab`** qui réalise l'allocation mémoire d'un tableau `tab` en fonction d'un entier donné représentant le nombre maximum d'éléments que peut contenir le tableau.
3. En utilisant la question précédente, écrire un algorithme nommé **`creatEnsVide`** qui réalise les tâches suivantes en fonction d'un entier donné `nMax` représentant le nombre maximum d'éléments que peut contenir l'ensemble :
   - Initialise le nombre d'éléments de l'ensemble à 0 ;
   - Affecte l'entier `nMax` au nombre maximum d'éléments de l'ensemble ;
   - Alloue la mémoire au membre tableau de l'ensemble en fonction de `nMax`.
4. Écrire un algorithme nommé **`estExist`** qui vérifie l'existence d'un élément donné dans un ensemble donné.

5. Ecrire un algorithme nommé **inserQueue** qui insère en queue un élément donné dans un ensemble donné.
   Note : *on suppose que l'ensemble est non plein et que l'élément à insérer n'existe pas dans l'ensemble.*
6. En utilisant les algorithmes `allocMemTab`, `estExist` et `inserQueue`; écrire un algorithme nommé **consEns** qui construit un ensemble à partir d'un tableau `tab` quelconque donné composé de n entiers.
   Exemple : à partir du tableau d'entier [13, 4, 1, 2, 13, 1, 0, 9, 0, 19, 12, 13, 1, 13] on obtient l'ensemble suivant {13, 4, 1, 2, 0, 9, 19, 12}.
   Note : *on utilise le nombre d'éléments présent dans le tableau pour la création de l'ensemble vide.*
7. En parcourant l'ensemble qu'une seule fois; écrire un algorithme nommé **adrMaxMin** qui détermine l'adresse du plus grand et du plus petit élément présent dans un ensemble non vide donné.
8. Compléter le programme C suivant :

```c
void main(void){
    TElement tab[DIM] = {13,4,1,2,13,1,0,9,0,19,12,13,1,13,4,1,2};
    Ensemble ens;
    int n;
    n = 17;
    // Faire appel a l'algorithme "consEns"
    ...
    // Faire appel a l'algorithme "adrMaxMin" pour afficher la valeur du
    // minimum et du maximum de l'ensemble
    ...
    // Afficher la valeur du voisin gauche du maximum s'il existe
    ...
    // Afficher la valeur du voisin droit du minimum s'il existe
    ...
}
```