UGA - UFR IM^{2}AG **Master 2 CCI**

Programmation et Langages 1

Devoir surveillé du 7 novembre 2025

Durée : 1h15 – Barème indicatif – **Documents autorisés : une feuille A4 recto-verso**

Exercice 1 ($\sim$ 5 points)

On considère le programme C suivant :

```
#include <stdio.h>

int x ; // variable globale

void f1(int x, int *y) {
  int a ;
  a = x ;
  *y = x + 1;
  x = a + 1 ;
  printf("a= %d, x=%d, *y=%d \n", a, x, *y) ;
}

int f2(int *x) {
  return *x + 1 ;
}

int main() {
  int a,b;
  int *p,*q;
  x = 42 ;
  p = &x ;
  q = &a;
  f1 (x, q) ;
  printf("a=%d, x=%d, *p=%d, *q=%d \n", a, x, *p, *q) ;
  *p = *q + 1 ;
  printf("a=%d, x=%d, *p=%d, *q=%d \n", a, x, *p, *q) ;
  b = f2 (p) ;
  printf("a=%d, b=%d, x=%d, *p=%d, *q=%d \n", a, b, x, *p, *q) ;
  return 0 ;
}
```

Q1. (4 points)

Donnez la séquence des affichages produits à l'écran lors de l'exécution de ce programme.

Q1. (1 point)

En considérant que les entiers et les pointeurs sont codés sur 4 octets, quelle est **la taille mémoire maximale** occupée par ce programme lors de son exécution, en tenant compte du fait que toutes les variables (et paramètres) déclarées dans le code source ne sont pas *actives* en même temps. Justifiez votre réponse en quelques lignes.

Exercice 2 (∼ 11 points)

Deux approches ont été vues en cours pour représenter des **séquences de longueur variable** en utilisant des **tableaux de tailles fixes** :

- soit en mémorisant explicitement la longueur de la séquence en plus de la séquence elle-même ;
- soit en indiquant implicitement cette longueur à l'aide d'une marque de fin

Dans le cas de séquences d'entiers positifs ces deux solutions peuvent être représentées par le lexique suivant (en notation algorithmique) :

lexique

{séquences avec longueur explicite}

SeqExp : le type < T : tableau sur [...] d'entiers positifs, L : un entier positif >

{séquences avec marque de fin}

M : la constante ... de type ... { Marque de fin }

SeqImp : le type tableau sur [...] d'entiers positifs

Q1. (2 points)

En considérant des séquence comportant **au plus** 256 éléments, traduisez ce lexique en langage C en complétant :

- la définition de la constante M ;
- les plages d'indices et tailles des tableaux utilisés [1].

Q2. (2 points)

Ecrivez en C les deux actions suivantes :

SeqExp2SeqImp : une action (la donnée S1 : une SeqExp, le résultat S2 : une SeqImp)
{copie S1 dans S2}

SeqImp2SeqExp : une action (la donnée S1 : une SeqImp, le résultat S2 : une SeqExp)
{copie S1 dans S2}

1. vous pouvez ajouter d'autres déclarations de constantes si vous le souhaitez

Q3. (2 points)

Ecrivez en C l'action suivante :
SommeBornéImp : une fonction (S : une SeqImp, n : un entier positif) : un entier positif
qui renvoie :

- la somme des n premiers éléments de S, si n est inférieur ou égal à la longueur de S
- -1 si n est strictement supérieur à la longueur de S

Q4. (2.5 points)

Ecrivez en C l'action suivante :
SommeBornéExp : une fonction (S : une SeqExp, n : un entier positif) : un entier positif
qui a les mêmes spécifications que la fonction SommeBornéImp.

Vous considèrerez les deux solutions suivantes pour écrire SommeBornéExp :

1. sans utiliser la fonction SommeBornéImp
2. en utilisant la fonction SommeBornéImp

Q5. (2.5 points)

En utilisant si besoin les fonctions déjà écrites, écrivez en C un **programme principal** qui implémente la suite d'instruction suivante :

1. déclare une variable Se de type SeqExp et une variable Si de type SeqImp ;
2. lit au clavier 100 valeurs entières et les mémorise dans Se ;
3. copie Se dans Si ;
4. affiche la somme des **50 derniers** éléments de Si.

Exercice 3 ($\sim$ 4 points)

On rappelle qu'en langage C les chaînes de caractères sont représentées dans des tableaux de caractères (de type `char`) et se terminent toujours par le caractère `'\0'` (marque de fin de chaîne). On rappelle également que chaque caractère est représenté par l'entier qui lui correspond dans le code ASCII et que, dans ce code, les lettres de l'alphabet sont numérotées de manières consécutives (de 'a' à 'z' pour les minuscules et de 'A' à 'Z' ppur les majuscules).

Ecrivez en C le corps de la fonction suivante :

```
void MinMaj (char *s) ;
// transforme en majuscule les caractères de s correspondant à des lettres minuscules,
//   les autres caractères restant inchangés.
```

Il n'est pas nécessaire de connaître le code ASCII, ni d'utiliser de fonctions de la bibliothèque `string.h`, pour répondre à cette question ...