GNG1506 Devoir 4

Le devoir doit être soumis par voie électronique avant la date d'échéance. La soumission doit être en format PDF. Pour les questions de programmation, il faut soumettre le fichier source (fichier.c). La solution à la question de la programmation doit être en format RAPPORT DE LABORATOIRE. Il faut inclure le fichier en formant PDF (excepté pour le code C qui est attaché séparément).

Question 1. Expliquer si le programme suivant aura une erreur de compilation, une erreur d'exécution ou une fuite de mémoire? Sinon, quel sera le résultat d'exécution?

```
#include <stdio.h>
int main ()
{
    int a[6]= {1 ,2 , 3, 4, 5, 6};
    printf ("%d\n", *(a +3));
    return 0;
}
```

Question 2. Expliquer si le programme suivant aura une erreur de compilation, une erreur d'exécution ou une fuite de mémoire? Sinon, quel sera le résultat de l'exécution?

```
#include <stdio.h>
int main ()
{
    int a [3]= {1 , 2, 3};
    int b [3]= {4 , 5, 6};
    a=b;
    for(i=0; i<3; i++)
        printf ("%d\n", a[i]);
    return 0;
}
```

Question 3. Expliquer si le programme suivant aura une erreur de compilation, une erreur d'exécution ou une fuite de mémoire? Sinon, quel sera le résultat de l'exécution?

```
#include <stdio.h>
int main ()
```

```
{
    int a[]= {1 , 2, 3, 4, 5, 6};
    int *p;
    p=a+3;
    printf ("%d\n", *(p+1));
    return 0;
}
```

Question 4. Expliquer si le programme suivant aura une erreur de compilation, une erreur d'exécution ou une fuite de mémoire? Sinon, quel sera le résultat de l'exécution?

```
#include <stdio.h>
int main ()
{
    int i;
    int a [6]= {1 , 2, 3, 4, 5, 6};
    int b[6]= { -1 , -2, -3, -4, -5, -6};
    *b=*a;
    for(i=0; i<3; i++)
        printf ("%d, %d\n", a[i], *(b+i));
    return 0;
}
```

Question 5. Dans le code ci-dessous, la fonction `func` nécessite un tableau de type `int` en entrée. Pour l'appel de `func`, la fonction `main` est utilisée en passant un pointeur (dans une certaine mémoire allouée). Est-ce que le code compile et fonctionne correctement? Qu'avez-vous appris de ce programme?

```
#include <stdio.h>
#include <stdlib.h>
void func (int a[], int lengthOfA )
{
    int i;
    i=0;
    for(i=0; i<lengthOfA; i++)
        a[i]=i;
}

int main ()
{
    int *b, i;
    b=( int *) malloc (sizeof (int )*5);
    if (b)
    {
```

```
        for(i=0; i<5; i++)
            b[i]=10+ i;
        func (b, 5);
        for(i=0; i<5; i++)
            printf ("%d\n", b[i]);
        free (b);
    }
    return 0;
}
```

Question 6. Expliquez qu'est ce qui ne va pas avec le programme suivant?

```
#include <stdio.h>
#include <stdlib.h>
void getIntegersFromUser (int N, int * userAnswers )
{
    int i;
    userAnswers =( int *) malloc (N* sizeof ( int ));
    if ( userAnswers )
    {
        printf (" Please enter %d integers \n", N);
        for(i=0; i<N; i++)
            scanf ("%d", userAnswers + i);
    }
}
int main ()
{
    int i, M=5;
    int *p;
    getIntegersFromUser (M, p);
    for(i=0; i<5; i++)
        printf ("%d\n", p[i]);
    return 0;
}
```

Question 7. Si un programme est écrit pour appeler la fonction `echange`, est-ce qu'il y aurait une erreur de compilation, potentiellement une erreur d'exécution ou une fuite de mémoire?

```
#include <stdio.h>
#include <stdlib.h>
void echange (int *a, int *b)
{
    int *tmp;
```

```
    tmp =( int *) malloc (sizeof (int ));
    *tmp =*a;
    *a=*b;
    *b=* tmp;
}
```

Question 8 (programmation). Écrire une fonction appelée `getEvenNumbers`. Le but de la fonction est de manipuler un tableau saisi en entrée de type `int` et de retourner la liste des nombres pairs dans le tableau. La fonction doit pouvoir manipuler des tableaux saisis avec des longueurs variables. Le type de retour de la fonction doit être "`int *`", ou un pointeur sur `int` (à savoir que la fonction rend un pointeur qui indique la mémoire stockant la liste des nombres pairs). On ne permet aucune variable globale. À part cette exigence, vous êtes libres de concevoir la fonction. Vous devez aussi écrire un programme de test (à savoir une fonction `main`) pour montrer le bon fonctionnement de votre conception et la mise en œuvre de cette fonction. Votre programme de test doit procéder les étapes suivantes. Premièrement, il demande à l'utilisateur d'entrer un entier positif N, ensuite il génère un tableau de longueur d'entiers aléatoires N, puis appelle la fonction `getEvenNumbers` pour obtenir la liste des nombres pairs du tableau et enfin il l'imprime.

Question 9 (programmation). Une matrice (un tableau rectangulaire de nombres) est appelé binaire si ses entrées sont 0 ou 1. Une matrice est appelée creuse (ou clairsemée) si la plupart de ses entrées sont des 0. Pour représenter une matrice creuse binaire, une approche plus efficace consiste à enregistrer les emplacements de toutes les entrées ayant une valeur 1 au lieu de le représenter comme un tableau à deux dimensions. L'objectif de cet exercice de programmation est de développer des matrices creuses binaires utilisant un pointeur sur des pointeurs et manipuler une telle représentation. Plus précisément, vous devez écrire un programme qui demande à l'utilisateur de saisir une matrice creuse binaire à partir du clavier (en entrant dans les emplacements de 1). Le programme imprime ensuite le nombre de 1 dans chaque colonne. Un exemple de sortie du programme est donné ci-dessous.

Entrez le nombre de lignes de la matrice:
[3]

Entrez le nombre de colonnes de la matrice:
[4]

Entrez le nombre de 1 dans la ligne 1:
[2]

Entrez les emplacements de colonne des 1 de la ligne 1:
[1]
[3]

Entrez le nombre de 1 dans la ligne 2:
[3]

Entrez les emplacements de colonne des 1 de la ligne 2:
[1]
[2]
[3]

Entrez le nombre de 1 dans la ligne 3:
[2]

Entrez les emplacements de colonne des 1 de la ligne 3:
[3]
[4]

Bien, j'ai obtenu la matrice:
Le nombre de 1 dans la colonne 1 est 2.
Le nombre de 1 dans la colonne 2 est 1.
Le nombre de 1 dans la colonne 3 est 3.
Le nombre de 1 dans la colonne 4 est 1.

Dans cet exemple, vérifiez que la matrice entrée par l'utilisateur est:

$$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

Ce programme doit être mis en œuvre par deux fonctions que vous devez concevoir. La première fonction est nommée `getBinarySparseMatrixFromUser` et doit avoir un type de retour `int **`. La deuxième fonction est nommée `printColumnStatisticsOfMatrix` et doit avoir un type de retour `void`. La fonction `main` doit avoir la structure suivante:

1. Déclaration des variables;
2. Appel getBinarySparseMatrixFromUser;
3. Appel printColumnStatisticsOfMatrix.

On ne permet aucune variable globale.