LANGAGE C – TP 6 et 7
Pointeurs

Exercice 3

On s'intéresse au code suivant :

```
/* TP6 - POINTEURS
SP - 08/21 - Exercice 3 */

/*--------------------------------*/
/* IMPORTATION BIBLIOTHEQUES      */
/*--------------------------------*/

#include <stdio.h>
#include <stdlib.h>

/*--------------------------------*/
/* PROGRAMME PRINCIPAL            */
/*--------------------------------*/

void main()
{
	int** matrice[4];
	int* ligne1[6];
	int* ligne2[6];
	int* ligne3[6];
	int* ligne4[6];
	ERRATUM :
	int** matrice[4] correspond à une matrice à trois dimensions.
	Pour rappel, les dimensions de ces structures sont équivalentes :
	int tableau[] et int* tableau
	int** tableau et int tableau[] et int tableau[][]
	Ici, on souhaite un tableau à 2 dimensions, donc int ** matrice
	ou int* matrice[4]
	Ensuite : les lignes sont des tableaux d'entiers donc les déclarations sont
	de la forme :
	int ligne1[6];
	int ligne2[6];
	int ligne3[6];
	int ligne4[6];

	for (int i = 0; i < 6; i++)
	{
		*(ligne1 + i) = i;
	}
	for (int i = 0; i < 6; i++)
	{
		*(ligne2 + i) = i*2;
	}
	for (int i = 0; i < 6; i++)
	{
		*(ligne3 + i) = i*3;
	}
	for (int i = 0; i < 6; i++)
```

```
	{
		*(ligne4 + i) = i*4;
	}
	matrice[0] = ligne1;
	matrice[1] = ligne2;
	matrice[2] = ligne3;
	matrice[3] = ligne4;
	for (int i = 0; i < 6; i++)
	{
		printf ("%d\t", *(ligne1 + i));
	}
	printf("\n");
	for (int i = 0; i < 6; i++)
	{
		printf("%d\t", *(ligne2 + i));
	}
	printf("\n");
	for (int i = 0; i < 6; i++)
	{
		printf("%d\t", *(ligne3 + i));
	}
	printf("\n");
	for (int i = 0; i < 6; i++)
	{
		printf("%d\t", *(ligne4 + i));
	}
	printf("\n\n");

}
```

Indiquer quelle sera la sortie écran.

Matrice serpent

Pour un entier naturel non nul n, on considère la matrice carrée $M(n)$ formée en serpent par les nombres $1, 2, 3, \dots, n^2$.

Par exemple :

$$M(2) = \begin{bmatrix} 1 & 2 \\ 4 & 3 \end{bmatrix} \qquad M(3) = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \\ 7 & 8 & 9 \end{bmatrix} \qquad M(4) = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 8 & 7 & 6 & 5 \\ 9 & 10 & 11 & 12 \\ 16 & 15 & 14 & 13 \end{bmatrix}$$

On a alors $\big(M(2)\big)_{21} = 4$.

1) Déterminer l'expression mathématique liant $\big(M(n)\big)_{ij}$, n, i et j.

On souhaite stocker une telle matrice dans un tableau à deux dimensions nommé `matriceSerpent`, où la dimension de la matrice est un entier demandé à l'utilisateur.

2) Ecrire une fonction de signature `int calculeElement(int i, int j, int n)` qui renvoie la valeur de l'élément `matriceSerpent[i][j]`.
3) Ecrire une fonction de signature `int** remplitSerpent(int n)` qui crée, remplit et retourne une matrice serpent de taille n.

```
int** remplitSerpent(int n){
   int ** matrice = (int**)malloc(n*sizeof(int*)); ERRATUM
   matrice est un tableau contenant n adresses de tableaux, chacune
correspondant à une ligne : donc (n*sizeof(int*)) et non (n*n*sizeof
(int*))
   assert( matrice != NULL);

   for (int i = 0; i < n; i++){
        int* ligne = (int*)malloc(n*sizeof(int));
        assert(ligne != NULL);
        matrice[i] = ligne;
   }

   for (int i = 0; i < n; i++){
        for (int j = 0; j < n; j++){
             matrice[i][j] = calculeElement(i, j, n);
        }
   }
   return matrice;
}
```

4) Ecrire une fonction de signature `void affiche(int **matrice, int n)` qui affiche la matrice passée en paramètre.
5) Ecrire une fonction de signature `void libereMatrice(int** M, int n)` qui libère l'espace mémoire alloué pour la matrice `matrice` de taille n.