

République Tunisienne	Section: Sc. Info	Niveau: 4^{ème} A
Ministère de l'Education	DS n° 3	Matière: Algo & Prog
CREs : Gabès-Médénine-Kebili & Tataouine	Durée: 3h	Date: 09-05-2024 à 8h

N.B. : Le sujet comporte **5 pages** numérotées de **1/5 à 5/5**
La page **1/5** est à remettre à la fin de l'épreuve avec la copie de l'examen

Exercice n°1 : (3 points)

Soit M une matrice carrée de type Mat ($n \times n$) entiers, avec Mat un tableau à deux dimensions et n un entier naturel impair, et soit la séquence algorithmique suivante :

```

S ← 0
Pour L de 0 à n-1 Faire
    Pour C de 0 à n-1 Faire
        Si condition Alors
            S ← S+M [ L,C]
        Fin Si
    Fin Pour
Fin Pour

```

Ecrire dans la colonne **Réponse** du tableau ci-dessus, la condition nécessaire permettant de calculer la somme des éléments hachurés S pour toute matrice carrée M de taille n.

	Eléments colorés	Réponse																																																																
1 ^{er} cas : n=5 S=91	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td>10</td><td>3</td><td>7</td><td>20</td><td>15</td></tr><tr><td>1</td><td>15</td><td>5</td><td>5</td><td>1</td><td>3</td></tr><tr><td>2</td><td>16</td><td>6</td><td>3</td><td>23</td><td>8</td></tr><tr><td>3</td><td>4</td><td>4</td><td>17</td><td>24</td><td>1</td></tr><tr><td>4</td><td>0</td><td>2</td><td>11</td><td>8</td><td>9</td></tr></table>		0	1	2	3	4	0	10	3	7	20	15	1	15	5	5	1	3	2	16	6	3	23	8	3	4	4	17	24	1	4	0	2	11	8	9																													
	0	1	2	3	4																																																													
0	10	3	7	20	15																																																													
1	15	5	5	1	3																																																													
2	16	6	3	23	8																																																													
3	4	4	17	24	1																																																													
4	0	2	11	8	9																																																													
2 ^{ème} cas : n=5 S=120	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>0</td><td>10</td><td>3</td><td>7</td><td>20</td><td>15</td></tr><tr><td>1</td><td>15</td><td>5</td><td>5</td><td>1</td><td>3</td></tr><tr><td>2</td><td>16</td><td>6</td><td>3</td><td>23</td><td>8</td></tr><tr><td>3</td><td>4</td><td>4</td><td>17</td><td>24</td><td>1</td></tr><tr><td>4</td><td>0</td><td>2</td><td>11</td><td>8</td><td>9</td></tr></table>		0	1	2	3	4	0	10	3	7	20	15	1	15	5	5	1	3	2	16	6	3	23	8	3	4	4	17	24	1	4	0	2	11	8	9																													
	0	1	2	3	4																																																													
0	10	3	7	20	15																																																													
1	15	5	5	1	3																																																													
2	16	6	3	23	8																																																													
3	4	4	17	24	1																																																													
4	0	2	11	8	9																																																													
3 ^{ème} cas : n=7 S=131	<table><tr><td></td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>0</td><td>10</td><td>3</td><td>7</td><td>15</td><td>5</td><td>2</td><td>4</td></tr><tr><td>1</td><td>15</td><td>5</td><td>0</td><td>3</td><td>8</td><td>4</td><td>3</td></tr><tr><td>2</td><td>16</td><td>6</td><td>3</td><td>8</td><td>0</td><td>11</td><td>5</td></tr><tr><td>3</td><td>4</td><td>4</td><td>22</td><td>1</td><td>5</td><td>8</td><td>7</td></tr><tr><td>4</td><td>0</td><td>2</td><td>11</td><td>9</td><td>0</td><td>7</td><td>12</td></tr><tr><td>5</td><td>8</td><td>14</td><td>20</td><td>8</td><td>7</td><td>5</td><td>21</td></tr><tr><td>6</td><td>10</td><td>6</td><td>7</td><td>2</td><td>1</td><td>0</td><td>16</td></tr></table>		0	1	2	3	4	5	6	0	10	3	7	15	5	2	4	1	15	5	0	3	8	4	3	2	16	6	3	8	0	11	5	3	4	4	22	1	5	8	7	4	0	2	11	9	0	7	12	5	8	14	20	8	7	5	21	6	10	6	7	2	1	0	16	
	0	1	2	3	4	5	6																																																											
0	10	3	7	15	5	2	4																																																											
1	15	5	0	3	8	4	3																																																											
2	16	6	3	8	0	11	5																																																											
3	4	4	22	1	5	8	7																																																											
4	0	2	11	9	0	7	12																																																											
5	8	14	20	8	7	5	21																																																											
6	10	6	7	2	1	0	16																																																											

Exercice n°2 : (3 points)

On souhaite calculer une valeur approximative de la constante π (pi) en utilisant les formules d'**Isaac Newton** et de **Nilakantha**, puis déterminer celle qui converge rapidement, à epsilon près vers π .

Formule de Newton :

$$\frac{\pi}{6} = \frac{1}{2} + \frac{1}{2} \left(\frac{1}{3 \times 2^3} \right) + \frac{1 \times 3}{2 \times 4} \left(\frac{1}{5 \times 2^5} \right) + \frac{1 \times 3 \times 5}{2 \times 4 \times 6} \left(\frac{1}{7 \times 2^7} \right) + \dots$$

Formule de Nilakantha :

$$\frac{\pi - 3}{4} = \frac{1}{2 \times 3 \times 4} - \frac{1}{4 \times 5 \times 6} + \frac{1}{6 \times 7 \times 8} - \dots$$

- 1) Ecrire un algorithme d'une fonction permettant de déterminer à quelle itération la formule de **Newton** atteint π à epsilon près.
- 2) Ecrire un algorithme d'une fonction permettant de déterminer à quelle itération la formule de **Nilakantha** atteint π à epsilon près.
- 3) Ecrire un module d'une fonction permettant d'afficher, selon le nombre d'itération, le **nom de la formule** qui converge à epsilon près vers π .

Exercice n°3 : (4 points)

En mathématiques, si **N** est un nombre décimal, **-N** est appelé l'opposé de **N** (-5 est l'opposé de 5). En binaire pour obtenir l'opposé de **N**, on utilise l'une de deux méthodes suivantes :

- **1^{ère} Méthode** : Elle est appelée **complément à 2 de N** : C'est une manière de représenter en binaire (sur 8 bits) les entiers négatifs.

Cette méthode consiste à **inverser** tous les 1 par 0 et tous les 0 par 1, puis on **ajoute 1**.

Exemple 1 :

Le nombre décimal **24** vaut en binaire : 11000

➔ Sur 8 bits on obtient : 00011000

➔ On inverse les bits : 11100111

➔ On ajoute 1 : 11100111+1 = 11101000

Donc -24 en binaire est : **11101000**

Exemple 2 :

Le nombre décimal **6** vaut en binaire : 110

➔ Sur 8 bits on obtient : 00000110

➔ On inverse les bits : 11111001

➔ On ajoute 1 : 11111001+1 = 11111010

Donc -6 en binaire est : **11111010**

👉 **N.B.** En binaire 0 + 0 = 0, 1 + 0 = 1 et 1 + 1 = 10

- **2^{ème} Méthode** : On garde tous les chiffres 0 depuis la droite jusqu'au premier 1 (compris) puis on inverse tous les suivants (à gauche).

Exemple 1 :

Le nombre décimal **24** vaut en binaire : 11000

- Sur 8 bits on obtient : 00011000

- Trouver le premier 1 depuis la droite : 00011000

- Inverser les restes : 11101000

Donc -24 en binaire est : **11101000**

Exemple 2 :

Le nombre décimal **6** vaut en binaire : 110

- Sur 8 bits on obtient : 00000110

- Trouver le premier 1 depuis la droite : 00000110

- Inverser les restes : 11111010

Donc -6 en binaire est : **11111010**

Travail demandé :

1. Écrire un algorithme une fonction nommée **Méthode1** permettant de déterminer l'opposé d'un nombre décimal donné et ceci en utilisant le principe de 1^{ère} méthode
2. Écrire un algorithme une fonction nommée **Méthode2** permettant de déterminer l'opposé d'un nombre décimal donné et ceci en utilisant le principe de 2^{ème} méthode

Exercice n°4 : (4 points)

Les chiffres **tomographiques** sont des méthodes pour chiffrer les messages.

Parmi ces méthodes on va utiliser la méthode du **chiffre de Chase** décrite ci-dessous :

1. On chiffre le mot **AMERIQUE** avec le chiffre de Chase. On commence par remplir un tableau de 3 lignes et de 10 colonnes par les lettres de l'alphabet. Les lignes du tableau sont numérotées 1,2,3 et les colonnes 0,1, 2, 3, ..., 9. Bien sûr, il y a 4 cases en trop, que l'on convient de remplir avec d'autres symboles, pour nous %, \$, &, +.

C'est ce tableau qui est la clé de chiffrement et qu'il convient d'avoir communiqué au préalable au destinataire du message.

Pour notre exemple, nous allons prendre la matrice **M** de type **Mat** suivante :

	0	1	2	3	4	5	6	7	8	9
1	%	X	U	A	C	O	N	L	Z	P
2	Q	B	Y	F	M	&	E	G	+	J
3	\$	D	K	S	V	H	R	W	T	I

2. On écrit verticalement, pour chaque lettre du message, ses coordonnées dans le tableau **T** de type **Tab**, en notant d'abord la ligne. On obtient le tableau suivant :

A	M	E	R	I	Q	U	E
1	2	2	3	3	2	1	2
3	4	6	6	9	0	2	6

3. On multiplie alors la deuxième ligne du tableau précédent par 9, c'est-à-dire que l'on multiplie **34669026** par **9** pour obtenir **312021234**. On revient alors à une forme utilisant des lettres, en reprenant la méthode initiale, mais dans l'ordre contraire, c'est-à-dire que l'on écrit :

	1	2	2	3	3	2	1	2
3	1	2	0	2	1	2	3	4
A	X	Y	Q	K	D	Y	A	M

4. La dernière lettre par exemple est **M**, car c'est la lettre que l'on trouve dans la ligne 2 et la colonne 4 du tableau. Seule la première lettre peut éventuellement poser un problème, ici on aurait pu choisir n'importe quelle lettre de la troisième colonne (**A** ou **F** ou **S**)

Travail demandé :

On dispose d'un fichier texte nommé "**source.txt**", contenant dans chaque ligne, un message à chiffrer. Chaque ligne du fichier texte contient des mots séparés par des symboles.

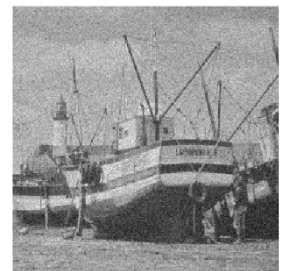
On vous demande d'écrire un algorithme du module permettant de chiffrer chaque ligne du fichier "**source.txt**" en respectant le principe décrit précédemment et mettre le résultat dans un fichier d'enregistrement nommé "**résultat.ch**" dont chaque enregistrement **E** de type **Enreg** est formé par les champs suivants : **Ch1**, **Ligne** et **Rang** de type respective **Chaine de caractères**, **Entier** et **Entier** où **Ch1** symbolise un mot crypté, **Ligne** représente son numéro de ligne dans le fichier texte, et **Rang** représente son ordre dans une ligne a crypté.

N.B.

- Un mot est formé uniquement par des lettres majuscules,
- Un symbole peut être soit : ?, %, \$, &, !
- Entre deux mots, on ne peut trouver qu'un seul séparateur.
- Un message ne peut ni débiter, ni se terminé par un symbole.

Exercice n°5 : (6 points)

On considère une image monochrome composée de 256 niveaux de gris, stockée dans une matrice de taille **L x C**, où **L** représente le nombre de lignes et **C** le nombre de colonnes. Chaque élément de cette matrice représente le niveau de gris d'un pixel de l'image, avec des valeurs allant de 0 à 255.



Cependant, les images peuvent parfois être de mauvaise qualité. Un exemple courant de défaut est le bruit, qui apparaît lorsque la photo est sous-exposée, c'est-à-dire qu'il y a un manque de luminosité. Ce bruit se manifeste par de petites fluctuations aléatoires des niveaux de gris, comme illustré dans la figure ci-dessous.

Pour éliminer le bruit des images, il est nécessaire d'appliquer une transformation aux valeurs des pixels. La méthode consiste à remplacer la valeur (a) de chaque pixel par la moyenne entre la valeur de (a) et des six pixels voisins en excluant la plus grande et la plus petite valeur parmi ses voisins.

c	d	e
b	a	f
i	h	g

Exemple :

54	47	192
79	190	153
203	189	166

Pour **a** = 190 → MaxVoisin=203, MinVoisin=47
Moyenne des 6 valeurs = $(79+54+192+153+166+189)/6$
= 138.83
La valeur **a** sera remplacée par la valeur $(190+138)/2=164$

54	47	192
79	164	153
203	189	166

En appliquant cette opération à chaque pixel (à l'exception des pixels des premières et dernières lignes ainsi que des premières et dernières colonnes), une partie du bruit est supprimée. En effet, ce bruit est caractérisé par des fluctuations aléatoires, qui sont atténuées par le calcul de moyennes. La figure ci-dessous illustre l'effet de cette opération.

Image bruitée



Image moyennée



Le problème à résoudre est le suivant :

Nous disposons d'un fichier texte nommé "**Img.txt**", qui représente les pixels d'une image monochrome. Ce fichier contient **L** lignes, chacune formée de **C** nombres entiers compris entre 0 et 255, séparés par des espaces. Les valeurs **L** et **C** sont toutes deux comprises entre 100 et 500. Entre deux nombres, on ne peut trouver qu'un seul espace.

Notre objectif est de réduire une partie du bruit en suivant le processus décrit précédemment.

Pour cela :

- Le contenu du fichier "**Img.txt**" sera stocké dans une matrice MAT de **L** lignes et **C** colonnes.
- Le traitement de suppression du bruit sera effectué sur la matrice.
- Ensuite, le contenu de la matrice sera réécrit dans le fichier "**Img.txt**", remplaçant ainsi son ancien contenu.

Travail demandé :

1. Écrire un algorithme du programme principal, solution à ce problème, en le décomposant en modules.
2. Écrire un algorithme pour chaque module envisagé.