جامعة مولاي إسماعيل
UNIVERSITÉ MOULAY ISMAÏL

كلية العلوم والتقنيات

Examen : Algorithmique et Programmation II
Module: I132
Responsables : F.AMOUNAS & B.OUCHAO
Durée : 3H

(3) (1)

Questions de cours

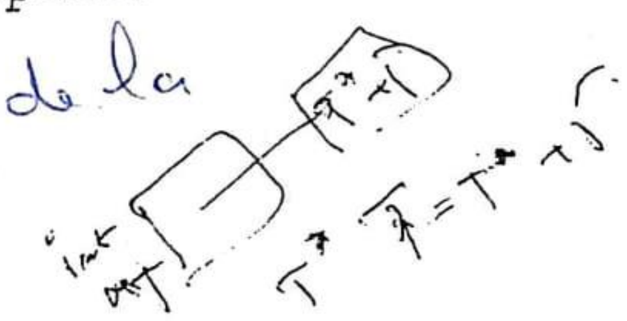
1) Indiquez les erreurs dans le code ci-dessous et proposez des corrections.

```
int nombre=12)
print("Donnez un nombre entier :") ;
scanf ("%d",nombre) ;
```

2) Qu'affiche le morceau de programme suivant? justifier votre réponse.

```
int *T;
int x = *T + 5;
printf ("%d", x) ;
```

il affiche le contenu de la variable X.



Exercice 1.

On considère un tableau T de N éléments distincts.

1. Ecrire une fonction « *Rang* » permettant de construire le tableau d'entiers R de même longueur tel que R[i] compte le nombre d'entrées du tableau T inférieures à T[i].

Exemple :

T

7	9	13	4	18	2	15	6
1	2	3	4	5	6	7	8

R

3	4	5	1	7	0	6	2
1	2	3	4	5	6	7	8

2. Déduire de cette construction un algorithme de tri de tableau T contenant N entiers distincts, on suppose que le résultat est représenté par le tableau TT.

Exercice 2.

On souhaite faire une simulation de vie des cellules. La structure utilisée est une grille (tableau à 2 dimensions) dans laquelle les cases peuvent prendre les valeurs 0 ou 1.

À chaque étape, on calcule la nouvelle valeur de chacune des cases en fonction des valeurs des cases voisines. Les cases voisines d'une case étant celles se trouvant de part et d'autre de cette case sur la même ligne ou sur la même colonne (pas en diagonale).

- La valeur d'une case passe à 0 si elle possède 0 ou 4 voisins à 1 (elle meurt isolée).
- La valeur d'une case passe à 1 si elle possède 2 ou 3 voisins à 1.
- Elle ne change pas dans les autres cas.

Pour éviter les cas particuliers des cases se trouvant en périphérie de la grille, on considérera que le tableau possède une bordure remplie de zéros (0).

Dans la figure représentée ci-dessous, la case à l'intersection de la 2ème ligne et de la 3ème colonne à 3 voisins à 1.

0	0	0	0	0	0
0	1	0	1	0	0
0	1	1	0	1	0
0	1	1	0	0	0
0	1	0	0	1	0
0	0	0	0	0	0

Etape 1 : créer un poi
T

Etape 2 : créer un
entier x qui reçoit la
valeur de T plus 1

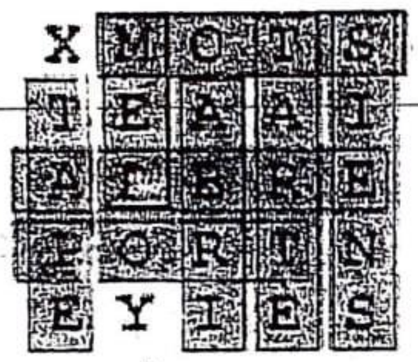
* T : permet d'accéder à la val

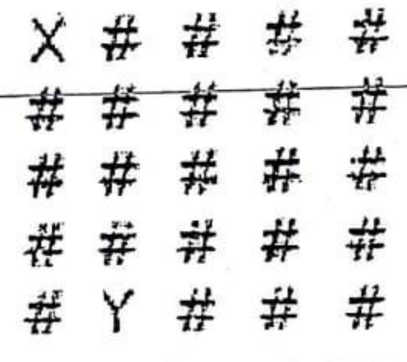
1. On souhaite pouvoir visualiser l'évolution de la grille. Écrivez une fonction «*Imprime*» qui affiche le contenu de la grille.
2. On souhaite pouvoir calculer pour une case donnée, le nombre de voisins à 1 afin de tester les différents cas. Écrivez une fonction « *nbVoisins* » permettant d'obtenir ce nombre.
3. On dispose d'une fonction Rand( ) qui retourne au hasard la valeur 0 ou la valeur 1. Écrire la fonction « *configInit* » qui permet d'initialiser la grille (bordure inclus).
4. Écrire la fonction « *estVivante* » qui retourne la valeur que devra prendre une case donnée à l'étape suivante.
5. Écrire la fonction « *changeEtat* » qui permet de faire évoluer la grille d'une étape vers une autre.

Exercice 3.

Une grille de "mots cachés" est un processus qui consiste, étant donnée un tableau de mots, à les rayer dans une grille de lettres afin d'isoler des lettres non utilisées (voir l'exemple).
Pour simplifier, les grilles sont carrées et les mots peuvent uniquement être lus de gauche à droite et de haut en bas, mais ils peuvent se chevaucher.
Exemple :

ABRI
ARBRE
MER
MOTS
PORT
SIENS
TAPE
TARTE





Le but de l'exercice est d'écrire un programme permettant de résoudre une grille de mots cachés en noircissant (afficher le caractère #) les mots de la liste dans la grille.

1. Quels sont les types en langage C du tableau de mots et de la grille ? Quelle est la différence ?
2. Ecrire la fonction « *newStringArray* » qui alloue l'espace mémoire nécessaire pour un tableau de *nb* chaînes de caractères et le renvoie.
3. Ecrire la fonction « *newCharMatrix* » qui alloue l'espace mémoire nécessaire pour une matrice de caractères de taille size × size et la renvoie.
4. Ecrire la fonction « *printMatrix* » qui affiche une matrice de caractères *mat* de taille size × size.
5. Ecrire la fonction « *lineIndices* » qui, étant donné un mot *word*, renvoie les indices entre lesquels il apparaît dans la ligne *line* de la matrice de caractères *mat* de taille size×size. Si le mot n'apparaît pas, la fonction renvoie une valeur impossible.
6. Ecrire une fonction « *solve* » qui, étant donnée une grille carrée de taille size×size et un tableau tab de *nb* chaînes de caractères, affiche une version de la grille de départ ou les mots du tableau sont remplacés par des #.

Bonne Chance