République Tunisienne **Ministère de l'éducation** **D-R-E MEDENINE** **LYCEES** **ROUTE DE GABES, IBN MEJEH, IBN SINA** **IBN KHALDOUN, MEDENINE, ABOU KACEM CHEBBI**	4ème année Sciences de l'informatique
	Algorithmique & programmation
	Date : Jeudi 12 Mai 2022 — Durée : **3** heures

DEVOIR DE SYNTHESE N° 3

Exercice 1 : 3 points

Un mot palindrome est une chaîne qui peut se lire indifféremment de gauche à droite ou de droite à gauche en gardant le même sens.

Exemple : *radar*, *rotor, ici*, *elle, non,…*

1. Ecrire l'algorithme d'une fonction récursive nommée **palindrome** qui vérifie si une chaîne donnée est palindrome ou non.
   - ⇨ Palindrome ("*radar") = Vrai*
   - ⇨ Palindrome ("*bac") = Faux*
2. On veut déterminer la plus longue sous_chaine palindrome à partir d'une chaine **ch** donnée

   Exemple : pour **ch** = "abcfcbc"

   ➔Les sous-chaines palindromes sont : "bcfcb" ⇨ Longueur = 5
   "cfc" ⇨ Longueur = 3
   "cbc" ⇨ Longueur = 3

   En utilisant la fonction **palindrome** précédente, Ecrire l'algorithme d'une fonction **plus_long_palindrome** qui retourne la plus longue sous-chaine palindrome à partir d'une chaine **ch**.

   - ⇨ plus_long_palindrome ("*abcfcbc") =* "*bcfcb"*

Exercice 2 : 3 points

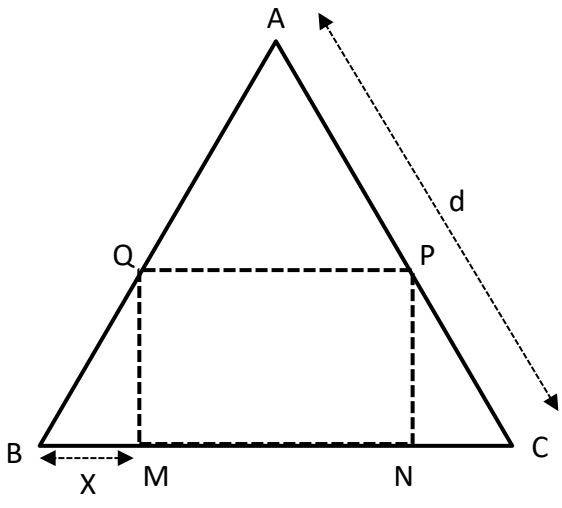
Soit **ABC** un triangle équilatéral dont le coté mesure **d** en cm. On inscrit dans ce triangle un rectangle **MNPQ**.

On pose **BM**=**X**

1) on donne tg $\frac{\Pi}{3} = \sqrt{3}$, Exprimer MQ en fonction de X

2) Exprimer la surface de rectangle MNPQ en fonction de x Sachant que MN=d – 2 *X

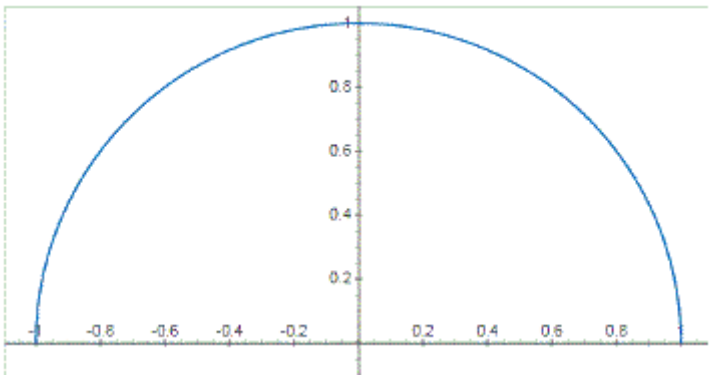
3) Ecrire l'algorithme d'une fonction **AireMaxi** qui retourne la valeur de **X** tel que l'aire du rectangle soit maximale pour une valeur **d** donnée.



Exercice 3 : 4 points

Soit la courbe représentative de la fonction f(x) = $\sqrt{1-x^2}$sur l'intervalle [-1, 1]

1) Ecrire un l'algorithme d'un module permettant de rechercher la valeur approchée du point fixe de la fonction f (f(x)=x) sur l'intervalle [0, 1] à epsilon prés. Epsilon est déjà saisie



2) On se propose de calculer une valeur approchée de Π en utilisant les deux formules suivantes :

Formule 1 :

$\Pi/2 = \int_{-1}^{1} f(x)dx$ avec f(x)= $\sqrt{1-x^2}$

Formule 2 :

$$\pi = 2\left[1 + \frac{1}{3} + \frac{1\times 2}{3\times 5} + \frac{1\times 2\times 3}{3\times 5\times 7} + \cdots\right]$$

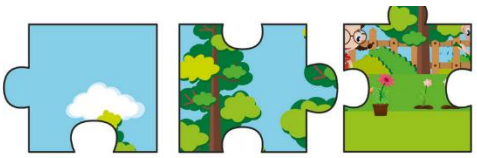
a) Ecrire un algorithme d'un module nommé **valeur_app1** qui permet de calculer une valeur approché de π à 10^{-4} prés en utilisant **la formule 1** et en utilisant la méthode de trapèze.

b) Ecrire un algorithme d'un module nommé **valeur_app2** qui permet de calculer une valeur approché de π à 10^{-4} prés en utilisant **la formule 2.**

c) Ecrire un algorithme d'un module nommé **comparaison** qui affiche parmi les deux formules précédentes celle dont la valeur approchée trouvée est la plus proche de la valeur d'une constante **Π = 3.1415**

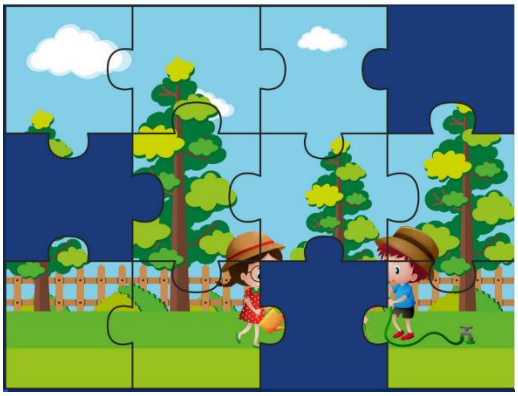
Exercice 4 : 5 points

Parmi les jeux populaires sur internet est celle de **Puzzle** qui consiste à reconstituer une image en utilisant **N** portions de cette image et les mettre à leurs bonnes places dans les cellules d'une matrice à **L** lignes et **C** colonnes (L*C=N).



Lorsque un utilisateur fini le jeu, sa réponse est envoyée à un serveur distant qui après vérification, lui envoi l'un des deux messages :

- ✓ *Bravo, passer au niveau suivant* si toutes les portions sont à leurs places.
- ✓ *C'est perdu, essayer de nouveau* en cas de perte.



Pour faciliter la résolution de ce problème on procède comme suit :

1. Ecrire l'algorithme d'un module **Saisie** qui saisit le nombre **N** de portions de l'image *(5≤N≤25)* et un nombre des lignes **L** et un nombre des colonnes **C**.
2. Ecrire l'algorithme d'un module **Remplissage** qui permet de remplir une matrice par les numéros des **N** portions de l'image d'une manière distincte.

   Exemple : pour une image contenant 12 portions, une matrice de 4x3 peut avoir le contenu suivant :

	0	1	2
0	5	9	6
1	2	10	8
2	7	3	4
3	12	1	11

3. Un fichier texte "**réponse.txt**" est généré automatiquement par l'ordinateur, contenant l'essai de joueur et il sera envoyé au serveur distant. Il qui contient sur chaque ligne :

   Numéro de la portion**:**indice ligne/indice colonne

   "réponse.txt"

```
5:0/0
9:0/1
6:0/2
2:1/0
10:1/1
8:1/2
7:2/0
3:2/1
4:2/2
12:3/0
1:3/1
11:3/2
```

   Ce fichier sera comparé à un fichier de données **"réponse_juste.dat"** ayant dans chaque enregistrement le numéro de la portion, sa position dans la matrice (indice ligne et indice colonne).

   Ecrire l'algorithme d'un module **Vérification** qui compare les deux fichiers et affiche le message apparu sur le Pc du joueur.

NB : vous n'êtes pas censé de remplir les deux fichiers "**réponse.txt**" et "**réponse_juste.dat**"

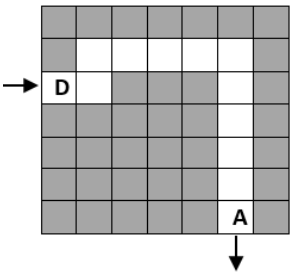
numP	indL	indC
6	0	0
3	1	0
4	2	2
....	...	....

"réponse_juste.dat"

Exercice 5 : 5 points

On modélise un labyrinthe par une matrice **M** carrée d'ordre **N**.
Dans cette matrice :

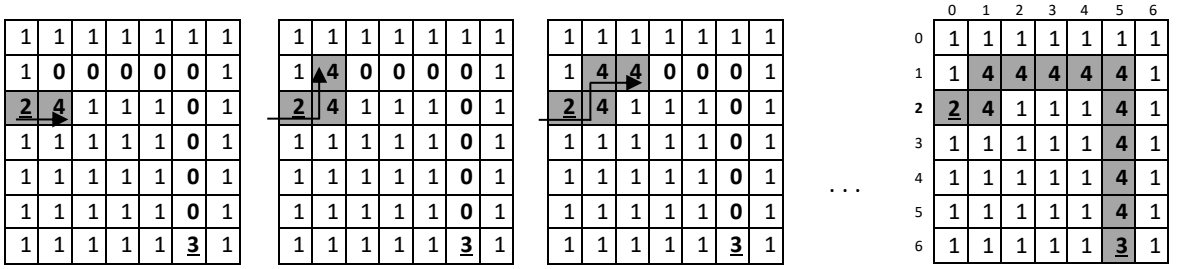
- 0 représente une case vide
- 1 représente un mur
- 2 représente le départ du labyrinthe
- 3 représente l'arrivée du labyrinthe



	0	1	2	3	4	5	N-1 6
0	1	1	1	1	1	1	1
1	1	**0**	**0**	**0**	**0**	**0**	1
2	**2**	**0**	1	1	1	**0**	1
3	1	1	1	1	1	**0**	1
4	1	1	1	1	1	**0**	1
5	1	1	1	1	1	**0**	1
N-1 6	1	1	1	1	1	**3**	1

m

On suppose que le labyrinthe possède un unique chemin allant du départ (colonne **0**) à l'arrivée (ligne **N-1**) sans repasser par la même case. Pour déterminer la solution de labyrinthe, on parcourt les cases vides (valeur=**0**) de proche en proche. Lors d'un tel parcours, afin d'éviter de tourner en rond, on remplace la valeur d'une case visitée dans la matrice par la valeur **4**.



	0	1	2	3	4	5	6
0	1	1	1	1	1	1	1
1	1	**4**	**4**	**4**	**4**	**4**	1
2	**2**	**4**	1	1	1	**4**	1
3	1	1	1	1	1	**4**	1
4	1	1	1	1	1	**4**	1
5	1	1	1	1	1	**4**	1
6	1	1	1	1	1	**3**	1

i , j
2 , 0
2 , 1
1 , 1
1 , 2
1 , 3
1 , 4
1 , 5
2 , 5
3 , 5
4 , 5
5 , 5
6 , 5

Chemin parcouru

Travail demandé :

1) Le départ d'un labyrinthe est toujours indiqué par une ligne de la colonne **0**. Ecrire l'algorithme d'une fonction **départ_lab** qui retourne le numéro de ligne de départ.
2) Soit la fonction **voisI** qui retourne le numéro de ligne de case suivante de chemin à parcourir à partir de case d'indice i et j

```
Fonction voisI (m : mat ; i, j, n : entier) : entier
Début
    P ← i
    Si (-1<i<n) et (-1<j<n) Alors
        Si (m[i+1,j]=0) ou (m[i+1,j]=3) Alors
            p ← i+1
        Sinon Si (m[i-1,j]=0) ou (m[i-1,j]=3) Alors
            p ← i-1
        Fin si
    Fin si
   Retourner p
Fin
```

Exécuter manuellement la fonction **voisI** pour les deux appels suivants avec **m** et **n** sont celles de l'exemple ci-dessus et la case de départ i=2, j=0 :

- voisI (m, 2, 0, n) = ……
- voisI (m, 2, 1, n) = ……

3) En se référant au fonction **voisI**, écrire une fonction **voisJ(m, i, j, n)** qui retourne le numéro de colonne de case suivante de chemin à parcourir à partir de case d'indice i et j
4) En utilisant les deux fonctions précédentes, écrire un module qui affiche le chemin à parcourir dans la matrice **m** de case de départ jusqu'à la case d'arrivée.