LABYRINTHE

Présentation du sujet :

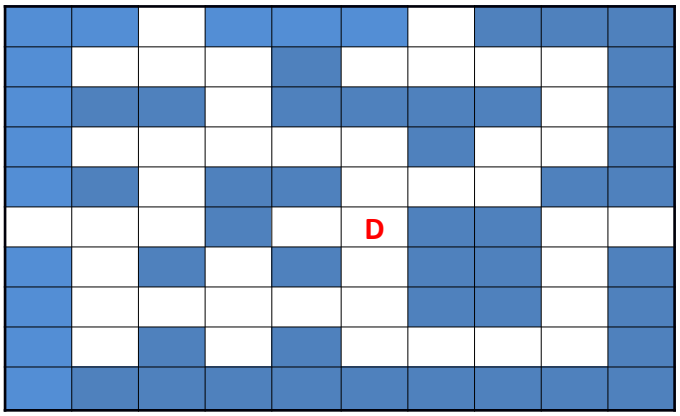
Un **labyrinthe** est tracé dans un carré de 100 cases (10 lignes et 10 colonnes).

La lettre D indique le point de départ.

L'objectif est de trouver comment atteindre l'une des sorties situées sur le bord du labyrinthe, le plus rapidement possible.

Le déplacement est possible vers le haut, vers la bas, vers la droite, ou vers la gauche, d'une case vide à une autre case vide (pas de déplacement en diagonale).

Le but de l'exercice est de calculer le nombre minimal de déplacements élémentaires pour sortir du labyrinthe.



PARTIE A :

Utilisation d'un ordinateur interdite. Les réponses sont à rédiger sur la feuille réponse (page 5)

Durée : 30 minutes

Un labyrinthe est défini par un tableau à 10 lignes et 10 colonnes. Chaque case du tableau contient une chaîne de caractères :

`' '` pour une case vide (espace)

`'X'` pour un mur.

`'0'` pour la case départ (zéro)

On suppose qu'une variable `Labyrinthe` de type tableau, est initialisée à :

```
Labyrinthe=[['X','X',' ','X','X','X',' ','X','X','X'],
           ['X',' ',' ',' ','X',' ',' ',' ',' ','X'],
           ['X','X','X',' ','X','X','X','X',' ','X'],
           ['X',' ',' ',' ',' ',' ','X',' ',' ','X'],
           ['X','X',' ','X','X',' ',' ',' ','X','X'],
           [' ',' ',' ','X',' ','0','X','X',' ',' '],
           ['X',' ','X',' ','X',' ','X','X',' ','X'],
           ['X',' ',' ',' ',' ',' ','X','X',' ','X'],
           ['X',' ','X',' ','X',' ',' ',' ',' ','X'],
           ['X','X','X','X','X','X','X','X','X','X']]
```

Les indices de ce tableau vont de 0 à 9 en lignes et en colonnes
On considère les procédures suivantes, définies par leur algorithme écrit en langage « naturel »
RAPPEL : La fonction `str` renvoie la chaîne de caractères associée à un entier, par exemple: `str(12)='12'`

```
procédure test_cases_adjacentes(i,j,n):

Si (i-1>=0):
        si Labyrinthe[i-1][j]=' ':
            Labyrinthe[i-1][j] ←str(n+1)
Si (j-1>=0):
        si Labyrinthe[i][j-1] =' ':
            Labyrinthe[i][j-1] ←str(n+1)
Si (i+1<10):
        si Labyrinthe[i+1][j]= ' ':
            Labyrinthe[i+1][j] ←str(n+1)
Si (j+1<10):
        si Labyrinthe[i][j+1] =' ':
            Labyrinthe[i][j+1] ←str(n+1)
fin procédure


procédure parcourir_lab(n):
    Pour i allant de 0 à 9:
        Pour j allant de 0 à 9:
            Si Labyrinthe[i][j]==str(n):
             alors teste_cases_adjacentes(i,j,n)
fin procédure
```

QUESTIONS:

Question A.1 :

a) Indiquer sur la feuille réponse le contenu de la variable Labyrinthe, après exécution des instructions successives : **test_cases_adjacentes(5,5,0)** puis **test_cases_adjacentes(4,5,1)**. On indiquera en couleur le contenu des cellules modifiées.

b) Expliquer à quoi servent les instructions conditionnelle suivantes dans la procédure teste_cases_adjacentes :

```
Si (i-1>=0)
Si (i+1<10)
```

Question A.2:

Ecrire l'algorithme d'une fonction `nombre_cases_vides(Labyrinthe)` qui renvoie le nombre de cases vides de la variable Labyrinthe, c'est à dire le nombre de cellules contenant la chaîne de caractère `' '`

Question A.3:

On suppose qu'on applique à la variable Labyrinthe (telle qu'elle a été définie *initialement*) les deux procédures :

`parcourir_lab(0)` puis **`parcourir_lab(1)`**

Indiquer sur la feuille réponse quel sera le contenu de la variable Labyrinthe après exécution de chacune de ces procédures.

PARTIE B :
Durée : 30 minutes

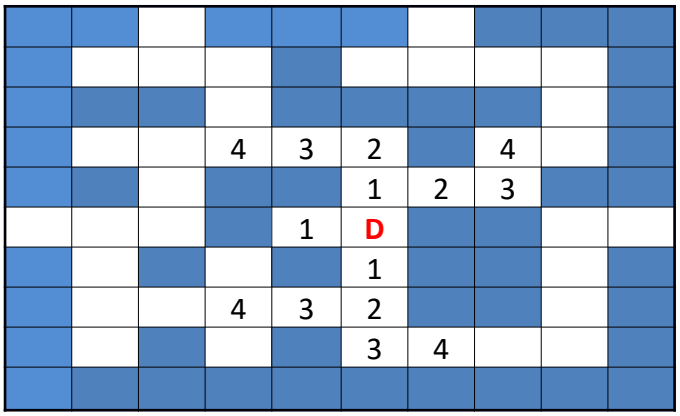
Pour cette partie, une implémentation de vos solutions algorithmiques est demandée. Pour cela vous travaillerez sur ordinateur. *En cas de difficulté d'implémentation, un algorithme langage naturel sera pris en compte dans l'évaluation. Dans ce cas, vous rédigerez votre algorithme sur une feuille de brouillon.*
Vous enregistrerez votre travail sur la clé USB dans un dossier à votre nom.
Ficher de travail : `Labyrinthe_A_COMPLETER.py`
Pensez à sauvegarder régulièrement votre travail.

Le but de cette partie est de repérer le chemin le plus court pour sortir du labyrinthe.

Question 1 :

a) Ecrivez un programme (en langage python) qui permet de remplir les cases vides du labyrinthe comme dans l'exemple ci-dessous, avec la chaîne de caractère donnant le nombre minimal de déplacements nécessaires pour atteindre cette case à partir de la case départ.

L'exemple ci-dessous l'illustre pour les cases accessibles en 4 déplacements maximum.



b) Modifier votre programme, de sorte que toutes les cases vides du tableau soient remplies.

Question 2 :

Pour aller plus loin, modifier le programme pour qu'il détermine (puis affiche) le nombre minimal de déplacements élémentaires nécessaires pour parvenir à sortir du labyrinthe.

C'est la longueur du « plus court chemin » qui part de la case départ et arrive à l'une des sorties du labyrinthe.

FEUILLE REPONSE DE LA PARTIE A

Question A.1 :

Labyrinthe initial

```
'X','X',' ','X','X','X',' ','X','X','X'
'X',' ',' ',' ','X',' ',' ',' ',' ','X'
'X','X','X',' ','X','X','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X',' ',' ','X'
'X','X',' ','X','X',' ',' ',' ','X','X'
' ',' ',' ','X',' ','0','X','X',' ',' '
'X',' ','X',' ','X',' ','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X','X',' ','X'
'X',' ','X',' ','X',' ',' ',' ',' ','X'
'X','X','X','X','X','X','X','X','X','X'
```

a) après l'instruction : test_cases_adjacentes(5,5,0)

```
'X','X',' ','X','X','X',' ','X','X','X'
'X',' ',' ',' ','X',' ',' ',' ',' ','X'
'X','X','X',' ','X','X','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X',' ',' ','X'
'X','X',' ','X','X',' ',' ',' ','X','X'
' ',' ',' ','X',' ','0','X','X',' ',' '
'X',' ','X',' ','X',' ','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X','X',' ','X'
'X',' ','X',' ','X',' ',' ',' ',' ','X'
'X','X','X','X','X','X','X','X','X','X'
```

puis après l'instruction test_cases_adjacentes(4,5,1).

```
'X','X',' ','X','X','X',' ','X','X','X'
'X',' ',' ',' ','X',' ',' ',' ',' ','X'
'X','X','X',' ','X','X','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X',' ',' ','X'
'X','X',' ','X','X',' ',' ',' ','X','X'
' ',' ',' ','X',' ','0','X','X',' ',' '
'X',' ','X',' ','X',' ','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X','X',' ','X'
'X',' ','X',' ','X',' ',' ',' ',' ','X'
'X','X','X','X','X','X','X','X','X','X'
```

b) Expliquer à quoi servent les instructions conditionnelle suivantes dans la procédure teste_cases_adjacentes :

```
Si (i-1>=0)

Si (i+1<10)
```

Question A.2:

Ecrire l'algorithme d'une fonction `nombre_cases_vides(Labyrinthe)` qui renvoie le nombre de cases vides de la variable Labyrinthe, c'est à dire le nombre de cellules contenant la chaîne de caractère `' '`

Question A.3:

On suppose qu'on applique à la variable Labyrinthe (telle qu'elle a été définie *initialement*) les deux procédures : `parcourir_lab(0)` puis `parcourir_lab(1)`

Indiquer sur la feuille réponse quel sera le contenu de la variable Labyrinthe après exécution de chacune de ces procédures.

Après `parcourir_lab(0)`

```
'X','X',' ','X','X','X',' ','X','X','X'
'X',' ',' ',' ','X',' ',' ',' ',' ',' ','X'
'X','X','X',' ','X','X','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X',' ',' ','X'
'X','X',' ','X','X',' ',' ',' ','X','X'
' ',' ',' ','X',' ','0','X','X',' ',' '
'X',' ','X',' ','X',' ','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X','X',' ','X'
'X',' ','X',' ','X',' ',' ',' ',' ','X'
'X','X','X','X','X','X','X','X','X','X'
```

Puis après `parcourir_lab(1)`

```
'X','X',' ','X','X','X',' ','X','X','X'
'X',' ',' ',' ','X',' ',' ',' ',' ',' ','X'
'X','X','X',' ','X','X','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X',' ',' ','X'
'X','X',' ','X','X',' ',' ',' ','X','X'
' ',' ',' ','X',' ','0','X','X',' ',' '
'X',' ','X',' ','X',' ','X','X',' ','X'
'X',' ',' ',' ',' ',' ','X','X',' ','X'
'X',' ','X',' ','X',' ',' ',' ',' ','X'
'X','X','X','X','X','X','X','X','X','X'
```