DS3 d'informatique (durée : 1h)

Le sujet vous demande d'écrire des fonction ou morceaux de programmes en Python, attention à indenter correctement votre code, vous pouvez marquer les différents niveaux d'indentation par des lignes verticales.

Si la syntaxe Python vous gène pour répondre à certaines questions, vous pouvez écrire la partie de programme qui pose problème en pseudo-code, c'est à dire en langue française, par exemple :

Si a est entier, alors ..., sinon ...

Pour i allant de 1 à 100 faire...

Exercice 1 : Carré magique

Un carré magique d'ordre n est composé de n^2 entiers strictement positifs, écrits sous la forme d'un tableau carré.
Ces nombres sont disposés de sorte que leurs sommes sur chaque ligne, sur chaque colonne et sur chaque diagonale soient égales.
On nomme alors constante magique la valeur de ces sommes.
Le carré magique est dit normal s'il est constitué de tous les nombres entiers de 1 à n^2.

1	6	8
5	7	3
9	2	4

Carré non magique

2	7	6
9	5	1
4	3	8

Carré magique normal d'ordre 3
Constante magique : 15

18	1	23
19	14	9
5	27	10

Carré magique d'ordre 3
Constante magique : 42

Le tableau sera stocké dans une liste de listes `T` telle que `T[i][j]` donne la valeur de l'élément de la ligne i et de la colonne j. Par exemple, le carré magique normal ci-dessus est stocké sous la forme :

```
T=[[2,7,6], [9,5,1],[4,3,8]]
```

L'objectif de l'exercice est d'écrire des fonctions qui permettent de vérifier si un tableau carré est magique ou pas, et, le cas échéant, de déterminer s'il est normal ou non. On s'interdira dans tout l'exercice l'utilisation de la fonction `sum`.

1. (a) Écrire une fonction `sommeLigne(T,i)` qui renvoie la somme des valeurs situées sur la ligne i du tableau `T`. On donnera la complexité de cette fonction en fonction de n.

   (b) Écrire de même une fonction `sommeColonne(T,j)` qui renvoie la somme des valeurs situées sur la colonne j du tableau `T` et deux fonctions `sommeDiag1(T)` et `sommeDiag2(T)` qui renvoient la somme des valeurs situées sur les diagonales du tableau `T`. Donner en fonction de n la complexité de ces fonctions.

2. Écrire une fonction `ListeSommes(T)` qui renvoie une liste constituée de toutes les sommes par ligne, par colonne et par diagonale des éléments du tableau `T`.
   On utilisera bien sûr les fonctions des questions précédentes.

3. Écrire une fonction `TousEgaux(L)` qui prend en paramètre une liste `L` et qui renvoie `True` si tous les éléments de `L` sont égaux et `False` sinon. On pourra supposer que `L` a au moins un élément. Donner la complexité de cette fonction en fonction de n.

4. Écrire une fonction `TousPresents(T)` qui renvoie `True` si tous les nombres de 1 à n^2 (où n est l'ordre du carré) sont présents dans `T`, et `False` sinon. Donner la complexité de cette fonction en fonction de n.
   Indication : il suffit de vérifier que tous les nombres sont entre 1 et n^2, et qu'ils ne sont présents qu'une fois chacun. On pourra pour cela utiliser une liste de booléens.

5. Écrire une fonction `magique(T)` qui affiche « Carré magique d'ordre ... et de constante magique ... » ou « Carré magique normal d'ordre ... et de constante magique ... » ou « Carré non magique » suivant la nature du tableau, en remplaçant les points de suspensions par les valeurs correspondantes comme dans les exemples ci-dessus. Exprimer en fonction de n la complexité de cette fonction.

Exercice 2 : La dichotomie

On souhaite résoudre numériquement une équation du type $f(x) = 0$, où $f(x)$ est une fonction continue. Plus précisément on cherche un nombre réel x_0 tel que $f(x_0) = 0$.

La méthode de dichotomie consiste à partir d'un intervalle $[a_0, b_0]$ sur lequel la fonction f change de signe et qui contient donc nécessairement une solution x_0. On coupe ensuite l'intervalle en deux et on conserve la partie $[a_1, b_1]$ sur laquelle la fonction f change de signe. En répétant cette opération on réduit progressivement la taille de l'intervalle contenant x_0. L'algorithme s'arrête lorsque la taille de l'intervalle $[a_n, b_n]$ est inférieure à la précision ε souhaitée sur le résultat.

1. Compléter la fonction python suivante qui détermine la valeur de x_0 par la méthode de dichotomie.

```
def zeroDichotomie(f,a,b,eps):
  while(....):
    c=....
    if f(c)==0:
      return c
    elif f(a)*f(c)<0 :
      b=c
```

```
    else:
      ....
  return (a+b)/2
```

2. Exprimer la taille de l'intervalle $l_n = b_n - a_n$ après n itérations en fonction de a_0, b_0 et n.
3. En déduire l'expression du nombre d'itérations nécessaire en fonction de a_0, b_0 et ε.

On souhaite résoudre l'équation

$$\sin(\cos(x)) = x^2$$

4. Écrire une fonction python `f1(x)` que l'on pourrait utiliser pour déterminer la solution de cette équation en utilisant la fonction `zeroDichotomie` ci-dessus.
5. Déterminer un intervalle de départ $[a_0, b_0]$ adéquat (On justifiera précisément pourquoi l'intervalle choisi convient). En déduire une estimation du nombre d'itérations nécessaire pour trouver une solution à 10^{-10} près.