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 : La dichotomie

On souhaite résoudre numériquement une équation du type $f(x) = 0$, où f 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 longueur 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(a)*f(c)<0 :
      ....
    else:
      ....
  return (a+b)/2
```

2. Montrer par récurence que la longueur de l'intervalle $[a_n; b_n]$ vaut $\frac{b_0-a_0}{2^n}$.
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 une solution de cette équation en utilisant la fonction `zeroDichotomie` ci-dessus. On supposera que toutes les fonctions de la bibliothèque `math` ont été importées.
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.

Exercice 2 : Quelques fonctions

Soit L une liste de n nombres entiers.

1. Écrire une fonction `plusGrand(L)` qui renvoie le plus grand élément de la liste L. Quelle est sa complexité en fonction de n ?
2. Écrire une fonction `somme(L)` qui renvoie la somme de tous les éléments de L. Quelle est la complexité en fonction de n ?

On donne la fonction suivante :

```
def truc(L, S):
  for i in range(len(L)):
    for j in range(len(L)):
      if L[i] + L[j] == S:
        return [i,j]
  return -1
```

3. Expliquer en quelques mots ce que fait la fonction `truc`.
4. Quelle est, en fonction de n, la complexité de la fonction `truc` dans le meilleur des cas ? Dans le pire des cas ?

Exercice 3 : Algorithme d'Euclide et PGCD

Le *P*lus *G*rand *C*ommun *D*iviseur (PGCD) de deux nombres a et b est le plus grand entier qui divise a et b simultanément. Par exemple le PGCD$(20, 30) = 10$, PGCD$(3, 8) = 1$ et PGCD$(4, 12) = 4$. On se propose d'étudier dans cet exercice l'algorithme d'Euclide pour déterminer le PGCD de deux nombres.

On commence par écrire une fonction qui effectue la division euclidienne de a par b et qui en renvoie le reste r. On suposera que $a \geqslant b > 0$.

```
def reste(a, b):
  r = a
  n = 0
  while r>=b:
    r = r-b
    n = n+1
  return r
```

1. Montrer que r est un *variant de boucle* et en déduire que la fonction se termine forcément.

2. Que représente la variable n dans ce programme ?
3. Montrer que $a = bn + r$ est un invariant de boucle, en déduire que la fonction précédente renvoie bien le reste de la division de a par b.

On traduit maintenant l'algorithme d'Euclide dans la fonction suivante :

```
def PGCD(A, B):
  a = A
  b = B
  r = reste(a, b)
  while r != 0:
    a = b
    b = r
    r = reste(a, b)
  return b
```

4. Décire le déroulement de l'appel de `PGCD(21, 15)`. On indiquera notamment les valeurs de a, b et r à la fin de chaque itération de la boucle.
5. Montrer que, dans la fonction précédente, r est un *variant de boucle*. En déduire que la fonction se termine.
6. Monter que si un nombre c divise simultanément a et b, il divise également b et $a - qb$. En déduire qu'un diviseur commun à a et b est aussi un diviseur commun à b et r.
7. En déduire que `PGCD(A,B) = PGCD(a,b)` est un invariant de boucle.
8. En utilisant la condition de sortie de la boucle, montrer que cette fonction renvoie effectivement PGCD(A, B).