Recherche d'un élément

Dans une liste

1. Écrire une fonction `recherche(x,L)` qui renvoie `True` si l'élément `x` apparait dans la liste `L` et renvoie `False` dans le cas contraire.

2. Vérifier que les instructions

```
print(recherche(4,[1,4,3,2]))
print(recherche(6,[1,4,3,2]))
```

affichent bien les résultats attendus.

3. Écrire une fonction `indice(x,L)` qui renvoie l'un des indices `i` pour lequel `L[i]=x`. Si un tel indice n'existe pas, on conviendra de renvoyer -1.

4. Vérifier que les instructions

```
print(indice(4,[1,4,3,2]))
print(indice(6,[1,4,3,2]))
```

affichent bien les résultats attendus.

5. Lorsque `x` apparait plusieurs fois dans la liste `L`, votr fonction `indice` renvoie-t-elle l'indice où `x` apparait pour la première fois ? ou pour la dernière fois ? ou un autre indice ?

Réponse : _____

6. Reprendre la fonction `indice` et la modifier de manière à avoir deux fonctions :
   - Une fonction `indicep(x,L)` qui renvoie le premier indice `i` tel que `L[i]=x` (et si un tel indice n'existe pas, renvoie -1) ;
   - Une fonction `indiced(x,L)` qui renvoie le dernier indice `i` tel que `L[i]=x` (et si un tel indice n'existe pas, renvoie -1).

7. Vérifier que les instructions

```
print(indicep(4,[1,4,3,4]))
print(indiced(4,[1,4,3,4]))
print(indiced(6,[1,4,3,4]))
```

affichent bien les résultats attendus.

Dans un tableau à deux dimensions

8. On considère dans la suite des tableaux de nombres à deux dimensions qui sont représentés par des listes de listes. On utilisera l'exemple suivant (à écrire dans votre code) :

```
M0 = [[1,-1,0,0,5],[2,3,1,-1,3],[1,1,1,-1,-1]]
```

qui représente le tableau à 2 dimensions :

$$M_0 = \begin{bmatrix} 1 & -1 & 0 & 0 & 5 \\ 2 & 3 & 1 & -1 & 3 \\ 1 & 1 & 1 & -1 & -1 \end{bmatrix}$$

Pour un tel tableau **M**, le coefficient en ligne i et colonne j est obtenu en écrivant **M[i][j]**, les numéros de ligne et colonne commençant à 0. La notation **len(M)** représente le nombre de lignes de **M** et **len(M[i])** représente le nombre d'éléments dans la i-ème ligne.
Tester le code suivant et comprendre à quoi correspond l'affichage :

```
def afficher(M):
    for i in range(0,len(M)):
        for j in range(0,len(M[i])):
            print(M[i][j])

afficher(M0)
```

9. Écrire une fonction **recherche2(x,M)** qui renvoie **True** si l'élément **x** apparait dans le tableau **M** et renvoie **False** dans le cas contraire.

10. Vérifier que les instructions

```
print(recherche(5,M0))
print(recherche(8,M0))
```

affichent bien les résultats attendus.

11. Écrire une fonction **indices2(x,M)** qui renvoie le couple **(i,j)** tel que **M[i][j]=x**. Si un tel couple n'existe pas, on conviendra de renvoyer $(-1,-1)$.

12. Vérifier que les instructions

```
print(indices2(5,M0))
print(indices2(8,M0))
```

affichent bien les résultats attendus.

13. Si l'élément **x** apparait plusieurs fois dans **M**, comment caractériser le couple **(i, j)** renvoyé par votre fonction **indice2** parmi tous ceux qui existent ?

Réponse : _____

Recherche d'un élément vérifiant une propriété

14. Écrire une fonction **premier_positif(L)** qui renvoie l'indice **i** du premier élément de **L** qui est positif (au sens large). On convient de renvoyer -1 si un tel élément n'existe pas.

15. Vérifier que les instructions :

```
print(premier_positif([1,2,3]))
print(premier_positif([-1,1,1]))
print(premier_positif([-1,-2]))
```

affichent bien les résultats attendus.

16. Écrire une fonction **dernier_positif(L)** qui renvoie l'indice **i** du dernier élément de **L** qui est positif (au sens large). On convient de renvoyer -1 si un tel élément n'existe pas.

17. Vérifier que les instructions :

```
print(dernier_positif([1,2,3]))
print(dernier_positif([-1,1,1]))
print(dernier_positif([-1,-2]))
```

affichent bien les résultats attendus.

18. Écrire une fonction **premier_changement_signe(L)** qui renvoie le premier indice **i** tel que **L[i]** et **L[i+1]** sont de signes contraires (au sens large) en convenant toujours de renvoyer -1 si un tel indice n'existe pas.

19. Vérifier que les instructions :

```
print(premier_changement_signe([1,2,3]))
print(premier_changement_signe([-1,1,1]))
print(premier_changement_signe([1,-1,1]))
```

affichent bien les résultats attendus.

Corrections

Q 1.

```
def recherche(x,L):
    for k in range(0,len(L)):
        if L[k]==x:
            return True
    return False
```

Q 2.

```
print(recherche(4,[1,4,3,2]))
print(recherche(6,[1,4,3,2]))
```

```
True False
```

Q 3.

```
def indice(x,L):
    for k in range(0,len(L)):
        if L[k]==x:
            return k
    return -1
```

Q 4.

```
print(indice(4,[1,4,3,2]))
print(indice(6,[1,4,3,2]))
```

```
1 -1
```

Q 6.

```
def indicep(x,L):
    for k in range(0,len(L)):
        if L[k]==x:
            return k
    return -1

def indiced(x,L):
    i = -1
    for k in range(0,len(L)):
        if L[k]==x:
            i = k
    return i
```

Q 7.

```
print(indicep(4,[1,4,3,4]))
print(indiced(4,[1,4,3,4]))
print(indiced(6,[1,4,3,4]))
```

```
1 3 -1
```

Q 8.

```
def afficher(M):
    for i in range(0,len(M)):
        for j in range(0,len(M[i])):
            print(M[i][j])

afficher(M0)
```

```
1 -1 0 0 5 2 3 1 -1 3 1 1 1 -1 -1
```

Q 9.

```
def recherche2(x,M):
    for i in range(0,len(M)):
        for j in range(0,len(M[i])):
            if M[i][j]==x:
                return True
    return False
```

Q 10.

```
print(recherche(5,M0))
print(recherche(8,M0))
```

```
False False
```

Q 11.

```
def indices2(x,M):
    for i in range(0,len(M)):
        for j in range(0,len(M[i])):
            if M[i][j]==x:
                return (i,j)
    return (-1,-1)
```

Q 12.

```
print(indices2(5,M0))
print(indices2(8,M0))
```

```
(0, 4) (-1, -1)
```

Q 14.

```
def premier_positif(L):
    for k in range(0,len(L)):
        if L[k]>=0:
            return k
    return -1
```

Q 15.

```
print(premier_positif([1,2,3]))
print(premier_positif([-1,1,1]))
print(premier_positif([-1,-2]))
```

```
0 1 -1
```

Q 16.

```
def dernier_positif(L):
    i = -1
    for k in range(0,len(L)):
        if L[k]>=0:
            i = k
    return i
```

Q 17.

```
print(dernier_positif([1,2,3]))
print(dernier_positif([-1,1,1]))
print(dernier_positif([-1,-2]))
```

```
2 2 -1
```

Q 18.

```
def premier_changement_signe(L):
    for k in range(0,len(L)-1):
        if L[k]*L[k+1]<=0:
            return k
    return -1
```

Q 19.

```
print(premier_changement_signe([1,2,3]))
print(premier_changement_signe([-1,1,1]))
print(premier_changement_signe([1,-1,1]))
```

```
-1 0 0
```