INFO-H-100 - Programmation
TP 9 — Quelques algorithmes

Lorsque l'exercice demande d'écrire une fonction, écrivez la fonction demandée et testez-la avec plusieurs valeurs pertinentes.

Ex. 1. Pour les exercices suivants, nous considérons qu'une matrice est représentée par une liste de listes de longueurs égales. Écrire une fonction `trace` à valeur réelle qui calcule la trace d'une matrice A de dimension $n \times n$ donnée en paramètre.

$$\texttt{trace}(A) = \sum_{i=1}^{n} A_{ii}$$

Ex. 2. Écrire une fonction qui compte le nombre de valeurs strictement négatives dans une matrice de nombres.

Ex. 3. Écrire une fonction qui détermine la somme des nombres positifs dans une matrice de nombres.

Ex. 4. Écrire les fonctions `symetrique` et `antisymetrique` qui testent respectivement la symétrie et l'antisymétrie d'une matrice carrée.

Pour rappel, une matrice A_{ij} ($i, j \in [0, n[$) est dite symétrique si elle est égale à sa transposée, c'est-à-dire si $\forall i, j : 0 \leq i, j < n : A_{ij} = A_{ji}$ et antisymétrique si $\forall i, j : 0 \leq i, j < n : A_{ij} = -A_{ji}$.

Ex. 5. Écrire une fonction qui effectue le produit de la matrice $A(l \times m)$ par la matrice $B(m \times n)$.

Ex. 6. Écrire une fonction qui, pour un n donné, affiche le triangle "Nord-Est" comme dans l'exemple ci-dessous où n vaut 5 :

```
* * * * *
  * * * *
    * * *
      * *
        *
```

Ex. 7. Écrire une fonction qui, pour un n donné, affiche le triangle "Nord-Est" numéroté comme dans l'exemple ci-dessous où n vaut 5 :

```
1 2 3 4 5
  1 2 3 4
    1 2 3
      1 2
        1
```

Vous pouvez, vous-même, inventer un très grand nombre de variations sur le thème de ces triangles, par exemple, en en changeant l'orientation ou la numérotation. Vous pouvez aussi, par exemple, écrire une fonction qui affiche un noeud-papillon de taille n donnée comme ci-dessous (pour $n = 7$) :

```
*           *
* *       * *
* * *   * * *
* * * * * * *
* * *   * * *
* *       * *
*           *
```

Ex. 8. Écrire une fonction `triSelection` qui trie de manière croissante une liste donnée en paramètre avec l'algorithme du tri par sélection. Par exemple :

```
ls = [2, 5, 9, 9, 6, 6, 4, 5, 2, 3, 1, 5, 8]
triSelection(ls)
print ls            # -> [1, 2, 2, 3, 4, 5, 5, 5, 6, 6, 8, 9, 9]
```

Ex. 9. Écrire une fonction `triInsertion` qui trie de manière croissante une liste donnée en paramètre avec l'algorithme du tri par insertion.

Ex. 10. Écrire une fonction `findDicho` qui détermine si, oui ou non, une valeur donnée existe dans une liste triée. Utiliser la recherche dichotomique.

INFO-H-100 - Programmation
TP 9 — Quelques algorithmes
Corrections

Solution de l'exercice 1:

```
def trace(mat):
    n = len(mat)
    res = 0
    for i in range(n):
        res += mat[i][i]
    return res
```

Solution de l'exercice 2:

```
def nbNeg(mat):
    """Le nombre de negatifs dans la matrice"""
    res = 0
    for l in range(len(mat)):
        for c in range(len(mat[l])):
            if mat[l][c] < 0:
                res += 1
    return res
```

ou

```
def nbNeg(mat):
    res = 0
    for ligne in mat:
        for val in ligne:
            if val < 0:
                res += 1
    return res
```

Solution de l'exercice 3:

```
def sommePositifs(mat):
    res = 0
    for ligne in mat:
        for val in ligne:
            if val > 0:
                res += val
    return res
```

Solution de l'exercice 4:

```
def symetrique(mat):
    n = len(mat)
    for l in range(n):
        for c in range(l + 1, n):
            if mat[l][c] != mat[c][l]:
                return False
    return True

def antiSymetrique(mat):
    n = len(mat)
    for l in range(n):
        for c in range(l, n):
            if mat[l][c] != -mat[c][l]:
                return False
    return True
```

ou, avec un seul return

```
def symetrique(mat):
    n = len(mat)
    res = True
    l = 0
    while l < n and res:
```

```
        c = l + 1
        while c < n and res:
            res = mat[l][c] == mat[c][l]
            c += 1
        l += 1
    return res

def antiSymetrique(mat):
    n = len(mat)
    res = True
    l = 0
    while l < n and res:
        c = l
        while c < n and res:
            res = mat[l][c] == -mat[c][l]
            c += 1
        l += 1
    return res
```

Solution de l'exercice 5:

```
def produit(mat1, mat2):
    res = []
    for l in range(len(mat1)):
        for c in range(len(mat2[0])):
            if c == 0:
                res.append([])
            x = 0
            for k in range(len(mat1[0])):
                x += mat1[l][k] * mat2[k][c]
            res[l].append(x)
    return res
```

Solution de l'exercice 6:

```
def triangleNordEst(n):
    for l in range(n):
        for c in range(l):
            print ' ',
        for c in range(l, n):
            print '*',
        print
```

ou

```
def triangleNordEst(n):
    for l in range(n):
        print '  ' * l + '* ' * (n - l)
```

Solution de l'exercice 7:

```
def triangleNordEstNum(n):
    for l in range(n):
        for c in range(l):
            print ' ',
        for c in range(0, n - l):
            print c + 1,
        print
```

Solution de l'exercice 8:

```
def posMinFrom(ls, i):
    """La position du min de ls en commencant en i.
    Pre: i dans ls
    """
    res = i
    while i < len(ls):
        if ls[i] < ls[res]:
            res = i
        i += 1
    return res

def echange(ls, i1, i2):
```

```
    ls[i1], ls[i2] = ls[i2], ls[i1]

def triSelection(ls):
    for i in range(len(ls) - 1):
        pos = posMinFrom(ls, i)
        echange(ls, i, pos)
```

Solution de l'exercice 9:

```
def deplacer(ls,i,j):
    """deplace, dans ls, la valeur en position i vers la position j
    Pre: i >= j
    """
    val = ls[i]
    del ls[i]
    ls.insert(j,val)

def posInsert(val, ls, n):
    """La position d'insertion de val parmi les n 1er elements de ls
    n si val est le plus grand
    """
    j = 0
    while j < n and val > ls[j]:
        j += 1
    return j

def triInsert(ls):
    for i in range(1, len(ls)):
        j = posInsert(ls[i], ls, i)
        deplacer(ls, i, j)
```

ou

```
def triInsert(ls):
    for i in range(1, len(ls)):
        val = ls[i]
        j = i
        while j > 0 and ls[j - 1] > val:
            ls[j] = ls[j - 1]
            j -= 1
        ls[j] = val
```

Solution de l'exercice 10:

```
def findDicho(ls, val):
    """True ssi val apparait dans ls
    Pre: ls est triee
    """
    bi = 0
    bs = len(ls) - 1
    m = (bs + bi) / 2
    while bi <= bs and ls[m] != val:
        if val < ls[m]:
            bs = m - 1
        else:
            bi = m + 1
        m = (bs + bi) / 2
    return bi <= bs
```