Programmation 2 Spring, 2026

TD 5: practice

1. Écrivez une fonction `longestConsecutive(A)` qui, étant donné une liste d'entiers `A` déjà triée par ordre croissant, renvoie la longueur de la plus longue sous-séquence d'éléments consécutifs dans la liste.

   Une sous-séquence d'éléments consécutifs signifie que les éléments se suivent dans la liste et que leurs valeurs sont des entiers consécutifs (c'est-à-dire que chaque élément est égal au précédent plus un).

   Vous ne pouvez pas utiliser de boucles imbriquées. Bien sûr, vous pouvez mettre des instructions conditionnelles à l'intérieur de la boucle, mais vous ne pouvez pas utiliser une boucle à l'intérieur d'une autre boucle.

   `Exemple :` Pour `A = [1,2,3,5,6,7,8,10]`, la fonction renvoie `4` car la séquence `[5,6,7,8]` est la plus longue suite déléments consécutifs (longueur 4).

   Pour `A = [10,11,12,13,14]`, la fonction renvoie `5` (tous les éléments sont consécutifs).

   Pour `A = [1,3,5,7]`, la fonction renvoie `1` (aucune paire d'éléments consécutifs).

   Solution.

```python
def longestConsecutive(A):
    _max, count = 1, 1
    for i in range(1, len(A)):
        if A[i] == (A[i-1] + 1):
            count += 1
        else:
            _max = max(_max, count)
            count = 1
    return _max

print( longestConsecutive( [1,2,4,5,6,7,11,12,17] ) )
```

2. Écrivez une fonction `findMedian_slow` qui prends une liste `A` d'entiers non triés, et renvoie la médiane dans `A`.

   `Exemple :`

   - Si `A = [3, 1, 4, 2]`, la médiane est 2 ou 3.
   - Si `A = [5, 2, 8, 1, 9]`, la médiane est 5 (élément du milieu après tri : `[1, 2, 5, 8, 9]`).

Solution.

```
def findMedian(A):
    n = len(A)
    for i in range(0, n):
        count = 0
        for j in range(0, n):
            if A[j] < A[i]:
                count += 1

        if count == n // 2:
            return A[i]
```

3. Écrivez une fonction `longestWord` qui prend une chaîne et renvoie le mot le plus long qu'elle contient. Vous ne devez prendre en compte aucune ponctuation lors du calcul de la longueur d'un mot.

   Example: pour `s = "Hello, how are you feeling today???"`, la fonction renvoie `"feeling"`.

Solution.

```
def longestWord(s):
    maxcount = 0
    maxword = ""

    currentcount = 0
    currentword = ""
    for i in range(0, len(s)):
        if s[i] == " " or i+1 == len(s):
            if currentcount > maxcount:
                maxcount = currentcount
                maxword = currentword
            currentcount = 0
            currentword = ""
        else:
            if (s[i] != "'" and s[i] != "?" and
                s[i] != "." and s[i] != "," and s[i] != "!") :
                currentcount = currentcount + 1
            currentword = currentword + s[i]
    return maxword


s = "Hello, how are you feeling today???"
print( longestWord(s) )
```

4. Écrivez une fonction `mysplit` qui prend une chaîne et renvoie une liste contenant tous les mots séparés par au moins un espace. Vous n'êtes autorisé à utiliser que les fonctions len et append.

   Example: pour `" asdf 8gas afbnn a "`, la fonction renvoie `['asdf', '8gas', 'afbnn', 'a']`.

Solution.

```
def mysplit(A):
    B = []
    start = 0
    end = 0

    i = 0
    while True:
        while i < len(A) and A[i] == " ":
            i = i + 1

        if i == len(A):
            return B

        start = i
        while i < len(A) and A[i] != " ":
            i = i + 1
        end = i
        B.append(A[start:end])

C = mysplit("   asdf 8gas afbnn   a ")
print(C)
```

5. Écrivez une fonction `partition` qui prend une liste d'entiers et une valeur `x`, et renvoie deux listes, l'une avec tous les éléments inférieurs ou égaux à `x` et l'autre avec tous les éléments supérieurs à `x`.

   Vous ne pouvez pas utiliser de boucles imbriquées. Bien sûr, vous pouvez mettre des instructions conditionnelles à l'intérieur de la boucle, mais vous ne pouvez pas utiliser une boucle à l'intérieur d'une autre boucle.

   `Example:` pour `[3,6,1,5,11,7,1], 5`, la fonction renvoie `[3,1,5,1] and [6,11,7]`.

Solution.

```
import random

def partition(A, x):
    S = []
    B = []
    for i in range(0, len(A)):
        if A[i] <= x:
            S.append(A[i])
        else:
            B.append(A[i])
    return S, B

A = [ random.randint(0,50) for _ in range(0, 10) ]
print("A: ", A, "\nB, C: ", partition(A, 20))
```

6. Écrivez une fonction `to_binary_string` qui reçoit un entier positif x en paramètre et retourne une chaîne de caractères représentant x en binaire.

   `Example:` pour `x = 10`, la fonction renvoie `'1010'`. Pour `x = 11`, la fonction renvoie `'1011'`.

Solution.

```
def to_binary_string(x):
    s = ''
    while x != 0:
        if x%2 == 1:
            s = '1' + s
        else:
            s = '0' + s
        x = x // 2
    return s

print( to_binary_string(11) )
```

7. Nous pouvons stocker un vecteur sous forme de liste de `floats` et une matrice sous forme de liste de listes de `floats`, où la i-ième liste représente la i-ième ligne de la matrice.

   Écrivez les fonctions suivantes :

   a) `addvectors` : renvoie la somme des deux vecteurs d'entrée

   b) `dotproduct` : renvoie le produit scalaire de deux vecteurs d'entrée

   c) `multiplyvectors` : renvoie le produit de la matrice d'entrée par un vecteur d'entrée

   d) `addmatrices` : renvoie la somme de deux matrices d'entrée

Solution.

```
def addvectors(u,v):
    if len(u) != len(v):
        return

    output = [0] * len(u)
    for i in range(0, len(u)):
        output[i] = u[i] + v[i]

    return output
```

```
def dotproduct(u,v):
    if len(u) != len(v):
        return

    output = 0
    for i in range(0, len(u)):
        output += u[i] * v[i]

    return output
```

```
def multiplyvectors(M,v):
    output = []
    for i in range(0, len(M)):
        if len(M[i]) != len(v):
            return
        val = 0
        for j in range(0, len(M[i])):
            val += M[i][j] * v[j]
        output.append(val)
    return output
```

```
def addmatrices(M1, M2):
    if len(M1) != len(M2):
        return

    output = []
    for i in range(0, len(M1)):
        output.append( [0]*len(M1[i]) )
        for j in range(0, len(M1[i])):
            output[i][j] = M1[i][j] + M2[i][j]
    return output
```

8. (**Difficile**.) Écrivez une fonction `squareNumbers` qui prend une liste d'entiers `A` triés par ordre croissant, et renvoie une liste des carrés de chaque nombre, également triés par ordre croissant.

   Vous ne pouvez pas utiliser de boucles imbriquées. Bien sûr, vous pouvez mettre des instructions conditionnelles à l'intérieur de la boucle, mais vous ne pouvez pas utiliser une boucle à l'intérieur d'une autre boucle.

   `Example:` pour `A = [-4,-1,0,3,10]`, la fonction renvoie `[0,1,9,16,100]`.

Solution 1.

```
def squareNumbers(A):
    pos = 0
    while pos < len(A) and A[pos] < 0:
        pos += 1

    neg = pos - 1

    B = []
    for _ in range(0, len(A)):
        if (pos >= len(A) or (neg >= 0 and pos < len(A) and A[neg]**2 <= A[pos]**2)):
            B.append(A[neg]**2)
            neg -= 1
        elif (neg < 0 or (neg >= 0 and pos < len(A) and A[neg]**2 > A[pos]**2)):
            B.append(A[pos]**2)
            pos += 1
        else:
            print("ERROR: this case should never happen!")

    return B


A = [-4,-1,0,3,10]
print( squareNumbers(A) )
```

Solution 2.

```
def squareNumbers(A):
    s, e = 0, len(A)-1
    B = [0]*len(A)

    i = len(A)-1
    while i >= 0:
        if A[e]**2 > A[s]**2:
            B[i] = A[e]**2
            e -= 1
        else:
            B[i] = A[s]**2
            s += 1
        i -= 1

    return B

A = [-4,-1,0,3,10]
print( squareNumbers(A) )
```

9. (**Difficile**.) On vous donne une liste `A` contenant n nombres déjà *triés* par ordre croissant.

   Écrivez une fonction `findTwoFast` qui prend `A` ainsi qu'un entier x et renvoie **deux indices distincts i et j** dans `A` telle que `A[i]+A[j] = x`. Elle renvoie `None` si cela n'est pas possible.

   Vous ne pouvez pas utiliser de boucles imbriquées. Bien sûr, vous pouvez mettre des instructions conditionnelles à l'intérieur de la boucle, mais vous ne pouvez pas utiliser une boucle à l'intérieur d'une autre boucle.

   Solution.

```
import random

def findTwoFast(A, x):
    s, e = 0, len(A)-1

    while s != e:
        if A[s]+A[e] == x:
            return s, e
        elif A[s]+A[e] < x:
            s += 1
        else:
            e -= 1
    return None

n = 20
x = 19
A = sorted([ random.randint(0,50) for _ in range(0, n) ])

print("A: ", A)
print("x: ", x)
print("findTwoFast: ", findTwoFast(A,x))
```