FSAC
FACULTE DES SCIENCES AIN CHOCK
UNIVERSITE HASSAN II DE CASABLANCA

TP 3 : STRUCTURES DE DONNEES

(*) Exercice facile
(**) Exercice de difficulté moyenne
(***) Exercice avancé

Partie I : Les listes & dictionnaires

Exercice 1 : Les polynômes

Un polynôme sera représenté par une liste, les valeurs sont les coefficients et les indices sont les poids.

Exemple : le polynôme suivant p(x) = $4x^3 + 2x + 6$ sera représenté par la liste :

P = [6, 2, 0, 4]

1. (*) Ecrire une fonction **poly(p, x)** qui calcule la valeur numérique du polynôme P pour x. (Vous pouvez utiliser un algorithme simple qui utilise la puissance ou la méthode de Horner)

$$P(x) = (((a_n x + a_{n-1})x + a_{n-2})x + ... + a_1)x + a_0$$

Exemple : poly(P, 2) = 42

```
def poly(P, x):
    result = 0
    for i in range(len(P)):
        result += P[i]*(x**i)
    return result

def horner(P, x):
    result = 0
    for c in P[::-1]:
        result = result*x + c
    return result
```

2. (*) Ecrire une fonction **somme(a, b)** qui calcule et retourne la somme de deux polynômes.

Exemple : pour les deux polynômes P = [5, 3, 2] (Représente $2x^2 + 3x + 5$) et Q = [4, 0, 1, 6] (Représente $6x^3 + 1x^2 + 4$)

somme(P, Q) retourne [9, 3, 3, 6], qui représente $6x^3 + 3x^2 + 3x + 9$.

```
def somme(P, Q):
    # Trouver le degré maximal
    n = max(len(P), len(Q))

    # Créer une liste pour le résultat avec des coefficients initiaux à 0
    resultat = [0] * n
```

```
    # Additionner les coefficients de P
    for i in range(len(P)):
        resultat[i] += P[i]

    # Additionner les coefficients de Q
    for i in range(len(Q)):
        resultat[i] += Q[i]

    return resultat
```

3. (**) Ecrire une fonction **produit(a, b)** qui calcule et retourne le produit de deux polynômes.

   Exemple : Pour P=[2, 0, 1, 2] et Q=[1, 2, 4] la produit(P,Q) retourne [2, 4, 9, 4, 8, 8]

```
def produit(a, b):
    # Degrés du polynôme résultat
    d = (len(a) + len(b) -1)
    r = [0]*d
    # Multiplier les polynômes
    for i in range(len(a)):
        for j in range(len(b)):
            r[i+j] = r[i+j] + a[i]*b[j]
    return r
```

Exercice 2 : Statistiques

1. (*) Ecrire une fonction, **moyenne(L)** qui retourne la moyenne de la liste L.

```
def moyenne(liste):
    return sum(liste) / len(liste) if liste else 0
```

2. (**) Ecrire une fonction, **mediane(L)** qui retourne la médiane de la liste L.

```
def mediane(liste):
    # trier la liste dans un ordre croissant
    sorted_liste = sorted(liste)
    n = len(sorted_liste)
    mid = n // 2
    if n % 2 == 0:
        return (sorted_liste[mid - 1] + sorted_liste[mid]) / 2
    else:
        return sorted_liste[mid]
```

3. (**) Ecrire une fonction, **mode(L)** qui retourne les modes de la liste L.

```
def mode(L):
    D = {}
    # compter les fréquences des éléments
    for i in range(len(L)):
```

```
        D[L[i]] = L.count(L[i])
    # chercher les modes
    mx = max(D.values())
    modes = []
    for k,v in D.items():
        if v==mx :
            modes.append(k)
    return modes
```

4. (***) Ecrire une fonction, **decomposer(n)** qui décompose un entier positif en nombres premiers.

   Exemple : decomposer(120) retourne [2, 2, 2, 3, 5]

```
def decomposer(n):
    L = []
    x = 2
    while n!=1 :
        while n%x == 0:
            L.append(x)
            n = n // x
        x += 1
    return L
```

Partie II : Les chaînes de caractères

1. (*) Implémentez une fonction **est_palindrome(s)** qui vérifie si la chaîne s est un palindrome, c'est-à-dire qu'elle se lit de la même manière dans les deux sens.

```
def est_palindrome(s) :
    s = s.lower()
    s = s.replace(' ', '')
    r = s[::-1]
    return s==r
```

2. (*) Implémentez une fonction **anagrams(s1, s2)** qui détermine si deux chaînes s1 et s2 sont des anagrammes, c'est-à-dire qu'elles contiennent les mêmes lettres avec la même fréquence, mais dans un ordre différent.

```
def anagrams(s1, s2):
    # Éliminer les espaces et mettre les chaînes en minuscules
    s1 = s1.replace(" ", "").lower()
    s2 = s2.replace(" ", "").lower()

    # Trier les caractères et comparer
```

```
    return sorted(s1) == sorted(s2)
```

3. (***) Écrivez une fonction **compresser (s)** qui prend une chaîne s composée de lettres répétées et la compresse en une forme plus concise. Par exemple, "aaabbcccc" deviendrait "a3b2c4".

```
def compresser(s):
    if not s:  # Si la chaîne est vide
        return ""

    # Initialiser la chaîne compressée et le compteur
    resultat = ''
    compteur = 1

    # Parcourir la chaîne
    for i in range(1, len(s)):
        if s[i] == s[i - 1]:
            compteur += 1  # Incrémente le compteur si le même caractère se répète
        else:
            # Ajoute le caractère et son compteur à la liste de résultats
            resultat += s[-1] + str(compteur)
            compteur = 1  # Réinitialise le compteur

    # Ajouter le dernier caractère et son compteur
    resultat += s[-1] + str(compteur)

    return resultat
```