Préparation n° 3

Opérations de base sur : Listes, Chaînes, Tuples
Complexité algorithmique

NB : Déterminer la complexité algorithmique de chaque fonction

1- Écrire la fonction **moyenne (L)** qui reçoit en paramètre une liste **L** de nombres, et qui retourne la moyenne des éléments de **L**. (La moyenne est la somme des éléments divisée par le nombre d'éléments)

2- Écrire la fonction **produit (L, x, y)** qui reçoit en paramètre une liste **L** de nombres, et deux nombres x et y tels que **x<=y**. La fonction retourne le produit des éléments de **L**, qui sont compris entre **x** et **y**.

3- Écrire la fonction **indice (x, L)** qui reçoit en paramètres une liste **L** et un élément **x**. la fonction retourne l'indice de **x** si **x** existe dans **L**, sinon, elle retourne **None**.

4- Écrire la fonction **inverser (L)** qui reçoit en paramètre une liste **L**, et qui retourne une liste contenant les éléments de **L** dans l'ordre inversée.

5- Écrire la fonction **extremums (L)** qui reçoit en paramètre une liste **L** de nombres, et qui retourne un tuple qui contient le plus grand et le plus petit élément de **L**.

6- Écrire la fonction **occurrence (x, L)** qui reçoit en paramètres une liste **L** et un élément **x**. la fonction retourne le nombre de fois que **x** apparait dans **L**.

7- Écrire la fonction **liste_ensemble (L)** qui reçoit en paramètre une liste **L**, et qui retourne **True** si la liste **L** ne contient pas de doublon, sinon, la fonction retourne **False**.

8- Écrire la fonction **sans_dbl (L)** qui reçoit en paramètre une liste **L**, et qui retourne une liste qui contient les différents éléments de **L**.

9- Écrire la fonction **list_occurrences (L)** qui reçoit en paramètre une liste **L**, et qui retourne une liste de tuples qui contiennent les éléments de **L**, et l'occurrence de chaque élément dans **L**.

Exemple : L= [29, 46, 29, 31, 55, 29, 29, 31, 29, 29]
La fonction **list_occurrences (L)** retourne la liste [(**46** , 1), (**55** , 1), (**29** , 6), (**31** , 2)]

10- Écrire la fonction **carree (S)** qui reçoit en paramètre une chaîne de caractères **S**, et qui retourne **True** si la chaîne **S** est carrée, sinon, la fonction retourne **False**. Exemple : 'chercher'

11- *Une chaîne de caractères est un palindrome si on peut la lire de gauche à droite, comme de droite à gauche*. Exemple : 'ressasser'

Écrire la fonction **palindrome (S)** qui reçoit en paramètre une chaîne de caractères **S**, et qui retourne **True** si la chaîne **S** est un palindrome, sinon, la fonction retourne **False**.

Problème

Q.1- Écrire la fonction **liste_puissance (L, p)** qui reçoit en paramètres une liste de nombres **L** et un entier positif **p**. La fonction retourne une nouvelle liste **R** qui contient les éléments de **L** élevés chacun à la puissance **p**.

Exemple : Liste_puissances ([4, -1, 3, 7, 2], 2) retourne la liste [16, 1, 9, 49, 4]

Q.2- Déterminer la complexité de la fonction liste_puissance(L, p)

Q.3- Écrire la fonction **puiss (x, k)** qui reçoit en paramètres un nombre **x** un entier positif **k**. La fonction retourne la valeur de $\mathbf{x^k}$, en utilisant le principe de **l'exponentiation rapide** suivant :

- si k =0 $\mathbf{x^k = 1}$
- si k est pair $\mathbf{x^k = (x^2)^{k/2}}$
- si k est impair $\mathbf{x^k = x*(x^2)^{(k-1)/2}}$

Q.4- Déterminer la complexité de la fonction puiss(x, k)

Q.5- Écrire la fonction **liste_puiss (L, n)** qui reçoit en paramètres une liste de nombres **L** et un entier positif **p**. La fonction retourne une nouvelle liste **T** qui contient les éléments de **L** élevés chacun à la puissance **n**, en utilisant le principe de **l'exponentiation rapide**.

Q.6- Déterminer la complexité de la fonction list_puiss(L, n)