Interro 13

Exercice 1.

1. Rappeler le genre du mot 'anagramme' et en donner une définition.
2. Donner le nombre d'anagrammes du mot 'anagramme'
3. Écrire une fonction `NbAppar(ch,a)` qui prend en argument une chaîne de caractères et un caractère et qui renvoie le nombre de fois que le caractère `a` apparait dans `ch`. Par exemple, `NbAppar('ANAGRAMME', 'A')` renvoie 3.
4. Soient `ch1` et `ch2` deux chaînes de caractères de même longueur. Donner une condition nécessaire et suffisante portant sur le nombre d'apparitions de chaque lettre pour que ces deux chaînes de caractères soient des anagrammes.
5. Écrire une fonction `VerfiAnag(ch1,ch2)` qui prend en argument deux chaînes de caractères et qui renvoie `True` si `ch1` et `ch2` sont des anagrammes et `False` sinon.
6. Parmi les quatre fonctions suivantes, déterminer l'unique fonction qui prend en argument une chaîne de caractèreS `ch` et un caractère `a` et qui renvoie la liste chaînes de caractères obtenue en insérant à chaque position possible de `ch`.

   Par exemple, pour `ch='BO'` et `a='A'`, on doit obtenir `['ABO', 'BAO', 'BOA']`. Pour `ch='BO'` et `a='B'`, on doit obtenir `['BBO', 'BBO', 'BOB']` (le premier et le deuxième 'BBO' correspondent respectivement à l'ajout du 'B' à la position 0 puis à la position 1)

```
def Inser1(ch, a) :
    L = []
    for k in range(len(ch)) :
        L.append(ch[:k]+a+ch[k:])
    return L
```

```
def Inser2(ch,a) :
    L = []
    for k in range(len(ch)+1) :
        L.append(ch[:k]+a+ch[k:])
    return L
```

```
def Inser3(ch,a) :
    L = []
    for k in range(len(ch)) :
        L.append(ch[:k]+a+ch[k+1:])
    return L
```

```
def Inser4(ch,a) :
    L = []
    for k in range(len(ch)+1) :
        L.append(ch[:k]+a+ch[k+1:])
    return L
```

7. Utiliser la fonction d'insertion précédente pour écrire une fonction `InserListe(L,a)` qui prend en argument une liste de chaînes de caractères et qui renvoie la liste des chaînes de caractères obtenues en insérant à chaque position possible de chacune des chaînes de caractères de `L`.

   Par exemple, `InserListe(['OB', 'BO'], 'B')` renvoie `['BOB','OBB','OBB','BBO','BBO','BOB']`.
8. Compléter la fonction `ListeAnag(ch)` qui prend en argument une chaîne de caractères afin qu'elle renvoie la liste des anagrammes de la chaîne de caractères éventuellement avec répétition.

   Par exemple, `ListeAnag('BOB')` devra renvoyer `['BOB','OBB','OBB','BBO','BBO','BOB']`.

```
    def ListeAnag(ch):
        L=['']
        n=...
        for i in range(n):
            L=inserListe(... , ...)
        return
```

9. Ecrire une fonction `SansRepet(L)` qui prend en argument une liste `L` pour et qui renvoie la même liste sans répétition.

   Par exemple, `SansRepet(['BOB','OBB','OBB','BBO','BBO','BOB']` renvoie `['BOB','OBB','BBO']`.
10. Utiliser les fonctions précédentes pour écrire une fonction `NbAnag(ch)` qui prend en argument une chaîne de caractères et qui renvoie le nombre d'anagrammes de ce mot.

Correction 1.

1. Genre et définition du mot “anagramme” Le mot *anagramme* est de genre féminin. Une anagramme est une permutation des lettres d'un mot qui forme un autre mot ou une suite de lettres.

2. Nombre d'anagrammes du mot “anagramme” Le nombre d'anagrammes d'un mot est donné par la formule :

$$\frac{n!}{n_1! \times n_2! \times \cdots \times n_k!}$$

où n est le nombre total de lettres et n_i est le nombre d'apparition de chaque lettre dans le mot.

Pour “anagramme”, avec les lettres A (3), N (1), G (1), R (1), M (2), E (1) :

$$\frac{9!}{3! \times 1! \times 1! \times 1! \times 2! \times 1!}.$$

3. Fonction `NbAppar(ch, a)` La fonction suivante compte le nombre d'apparitions d'un caractère `a` dans une chaîne `ch` :

```
def NbAppar(ch, a):
    c=0
    for lettre in ch:
        if lettre == a:
            c+=1
    return c
```

Exemple : `NbAppar('ANAGRAMME', 'A')` renvoie 3.

4. Condition pour que deux chaînes soient des anagrammes Deux chaînes `ch1` et `ch2` de même longueur sont des anagrammes si et seulement si elles contiennent le même nombre d'occurrences pour chaque lettre.

5. Fonction `VerifAnag(ch1, ch2)` La fonction suivante vérifie si deux chaînes sont des anagrammes :

```
def VerifAnag(ch1, ch2):
    for lettre in ch1:
        if NbAppar(ch2,lettre) !=NbAppar(ch1,lettre):
            return False
    return True
```

6. Fonction d'insertion correcte La fonction correcte pour insérer un caractère `a` à chaque position possible dans une chaîne `ch` est :

```
def Inser2(ch, a):
    L = []
    for k in range(len(ch) + 1):
        L.append(ch[:k] + a + ch[k:])
    return L
```

7. Fonction `InserListe(L, a)` La fonction suivante applique `Inser2` à chaque élément d'une liste `L` :

```
def InserListe(L, a):
    result = []
    for ch in L:
        result.append(Inser2(ch, a))
    return result
```

8. Fonction `ListeAnag(ch)` La fonction `ListeAnag` génère les anagrammes d'une chaîne :

```
def ListeAnag(ch):
    L = ['']
    for lettre in ch:
        L = InserListe(L, lettre)
    return L
```

9. Fonction `SansRepet(L)` La fonction suivante élimine les doublons d'une liste :

```
def SansRepet(L):
    L2=[]
    for l in L:
        if l not in L2:
            L2.append(l)
    return L2
```

10. Fonction `NbAnag(ch)` Pour calculer le nombre d'anagrammes uniques d'une chaîne :

```
def NbAnag(ch):
    return len(SansRepet(ListeAnag(ch)))
```