EXERCICE 5 (4 points)

Cet exercice porte sur la programmation de base en Python et sur la récursivité.

Le Scrabble est un jeu de société qui se joue à plusieurs joueurs. Chaque joueur pioche des jetons sur lesquels sont inscrites une lettre majuscule et sa valeur.

Le joueur propose ensuite un mot qu'il dispose sur un plateau de jeu en plaçant un jeton par case.

Certaines cases permettent une bonification.

1. Lorsqu'on pose un mot sur des cases ne présentant aucune bonification, on calcule le score en additionnant la valeur des lettres qui composent le mot.

J 8	E 1	U 1

vaut 10 points.

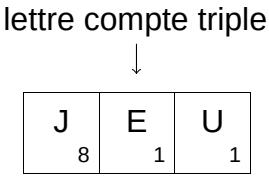
Voici le code d'une fonction *calculer_score_sans_bonif* qui a pour paramètre une chaîne de caractères mot et qui renvoie le score du mot lorsqu'il n'y a pas de bonification.

```
def calculer_score_sans_bonif(mot) :
1     scrabble = {"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,
2             "H":4, "I":1, "J":8, "K":10,"L":1,"M":2,"N":1,"O":1,
3             "P":3, "Q":8, "R":1,"S":1,"T":1,"U":1,"V":4,"W":10,
4             "X":10,"Y":10, "Z":10}
5     score = 0
7        for k in range(len(mot)) :
8    score = scrabble[mot[k]]
9    return score
```

Une erreur s'est glissée dans le corps de la fonction entre les lignes 5 et 9 incluses.
Indiquer la ligne à modifier et corriger l'erreur.

2. Sur certaines cases est indiquée la bonification "lettre compte double" ou "lettre compte triple".

Exemple :



Le score obtenu est de 12.

La fonction *calculer_score_avec_bonif* a pour paramètres :

- une chaîne de caractères mot
- une liste bonif de même longueur que mot précisant pour chaque lettre la bonification correspondante.

Cette fonction renvoie le score du mot en tenant compte des bonifications.

Dans l'exemple précédent on appelle *calculer_score_avec_bonif* ("JEU", [1, 3, 1]) et on obtient 12.

```
1 def calculer_score_avec_bonif(mot, bonif) :
2     scrabble={"A":1,"B":3,"C":3,"D":2,"E":1,"F":4,"G":2,
3    "H":4, "I":1, "J":8, "K":10, "L":1,"M":2,"N":1,"O":1,
4     "P":3, "Q":8, "R":1,"S":1,"T":1,"U":1,"V":4,"W":10,
5     "X":10,"Y":10, "Z":10}
6     score = 0
7     for k in range(len(mot)) :
8          score = # A compléter
9     return score
```

Recopier et compléter la ligne incomplète.

3. Lucien joue une partie avec ses grands-parents. Avec le tirage B A O L E L N, il a composé une liste de mots : BAL, NOBLE, ANE, BALLE, LOBE. Il souhaite mettre la lettre B sur une case "lettre compte triple", mais pour cela le B doit être la 3e lettre du mot qu'il doit poser.

(a) Ecrire une fonction *mot_lettre_position* qui a pour paramètres :

- une liste de mots liste
- un caractère c
- une position i (la position 2 correspond à la 3e lettre)

et qui renvoie la liste de tous les mots qui sont dans liste et qui ont le caractère c en position i.

(b) Quel appel Lucien doit-il effectuer pour trouver les solutions à son problème?

4. Pour passer le temps entre deux tours, Lucien décide de chercher les anagrammes d'un mot. L'anagramme d'un mot est un nouveau mot formé en changeant de place les lettres du mot initial.

Ainsi CARTE est une anagramme du mot ACTER.

On se propose d'écrire une fonction récursive *anagramme* qui a pour paramètres deux chaînes de caractères mot1 et mot2 et qui renvoie True si mot2 est une anagramme de mot1 et False sinon.

Pour cela, on dispose d'une fonction *enlever* :

```
enlever (mot , c)  renvoie mot privé de la première
occurence du caractère c
```

Ainsi l'instruction `enlever('scrabble','b')` renvoie 'scrable'.

```
1   def anagramme(mot1, mot2):
2     if len(mot1) != len(mot2):
3         return # A compléter
4     elif len(mot1) == 0:
5         return # A compléter
6     elif not mot1[0] in mot2 :
7         return # A compléter
8     else :
9         # Partie récursive à compléter
```

Recopier et compléter la fonction récursive anagramme.