NOM: **PRENOM:**

La clarté et la concision seront prises en compte lors de la correction. Vous devez impérativement respecter les consignes suivantes :

- Vous téléchargez le fichier `http://www.irif.fr/~haberm/sol_502188_82.py`
- Vous remplissez votre nom et prénom dans le fichier.
- Vous renommez ce fichier et rendez un unique fichier `Prenom_Nom.py`.
- Au début vous pouvez importer les modules dont vous aurez besoin.
- Les noms de fonctions que vous écrirez **doivent rester** identiques à ceux du sujet et vous modifiez **uniquement** le corps des fonctions.
- Laissez les symboles `#` en place et n'en utilisez pas ailleurs.
- Si vous avez besoin de fonctions intermédiaires, définissez les **à l'intérieur** de la fonction demandée.
- Les fonctions ne doivent rien **afficher** (pas d'appel de la fonction `print`).
- Exemple:

```
"""...."""
import string
def question1c(l):
    def mafonctionauxilliaire():
        return
    mafonctionauxilliaire()
    return l
#
def ....
#
```

1 Écrire une fonction `question1e` qui prend comme argument une liste et qui renvoie le produit du 2ème et du dernier élément. Par exemple `question1e([1,2,3,4,5,6,7,8,9,10])` renvoie 20. On suppose que la liste est suffisamment longue.

2 Écrire une fonction `question1d` qui prend comme argument une liste et qui renvoie le produit du 3ème et du dernier élément. Par exemple `question1d([1,2,3,4,5,6,7,8,9,10])` renvoie 30. On suppose que la liste est suffisamment longue.

3 Écrire une fonction `question2c` qui prend comme argument une liste d'entiers positifs et qui renvoie la liste des entiers de la liste d'entrée qui sont supérieurs ou égaux à 7. Par exemple pour `[10,9,2,3,4,5,6,7,8]` on obtient `[10,9,7,8]`.

4 Écrire une fonction `question3a` qui prend comme argument une chaîne de caractères composée de chiffres et qui renvoie une liste de tous ses chiffres divisibles par 3 auxquels on a ajouté 2. Par exemple pour `"124567234689"` on obtient `[8,5,8,11]`.

5 Écrire une fonction `question4e` qui prend en argument une liste de listes d'entiers et qui renvoie une liste de toutes les listes de longueur inférieure ou égale à 3 où tous les éléments ont été multipliés par 3. Par exemple pour `[[1,2],[2,3,4,5],[3,4,5]]` on obtient `[[3,6],[9,12,15]]`.

6 Écrire une fonction `question5d` qui prend en argument le nom d'un fichier et qui retourne le nombre de fois où le fichier contient la lettre 'd' ou la lettre 'a'. Par exemple pour le fichier `test5` sur Moodle contenant:

```
aazefo ioaze fi bwbblariufffdddqqfff
qfg  uiuuaaabbeecccd dedeee
```

`question5d("test5")` retourne `13`.

7 Écrire une fonction `question6b` qui prend en argument le nom d'un fichier et retourne un dictionnaire de mots où chaque clé est une lettre de l'alphabet et sa valeur correspondante la liste des mots du fichier commençant par cette clé. Le fichier considéré comportera des mots sans lettre accentuée, et les mots seront séparés par un espace sans aucun retour à la ligne. Par exemple, pour un fichier contenant

```
pierre penelope bonjour bazar truc
```

on obtient `{'t': ['truc'], 'b': ['bonjour', 'bazar'], 'p': ['pierre', 'penelope']}`
Sur Moodle vous trouverez un fichier long `motsquestion6b` pour tester.

8 Écrire une fonction `question7d` qui prend en argument une chaîne de caractères et qui renvoie la première suite consécutive de 6 voyelles ('a', 'e', 'i', 'o', 'u' ou 'y') qui s'y trouvent ou `None` s'il n'y en a pas. Par exemple pour `"abebgeayuaieebbioaa"` on obtient `'eayuai'`. Utilisez les expressions régulières.

9 Écrire une fonction `question8b` qui prend en argument un ensemble de chaînes de caractères et qui renvoie une liste de toutes les chaînes qui contiennent la **même** suite de 3 chiffres (entre 0 et 9) au moins 2 fois. Par exemple, pour `{"1a24bbb24a24","00000","23032311230023","111111"}` on obtient `['111111','23032311230023']`. Utilisez les expressions régulières.

10 La RATP fournit des données sur la fréquentation annuelle des stations de son réseau. Le fichier CSV (sur Moodle: `ratp.csv`) a le format suivant
`Réseau;Station;Trafic;Correspondance_1;Corr_2;Corr_3;Corr_4;Corr_5;Ville;Arr. pour Paris`
Par exemple la première ligne du fichier
`Métro;SAINT-LAZARE;45309544;3;9;12;13;14;Paris;8`
indique que le trafic annuel à la station Saint-Lazare à Paris (accès aux lignes 3,9,12,13 et 14) dans le 8e arrondissement de Paris est de 45309544 passagers. S'il y a moins que 5 lignes accessibles à une station les champs correspondants sont vides.

Écrire une fonction `question9c` qui prend en entrée un nom de fichier CSV du format donné ci-dessus et retourne la liste (sous forme de liste python) des lignes (chaînes de caractères) du fichier (sans le `\n`) concernant les stations de **Métro à l'extérieur de Paris**.

11 Écrire une fonction `question10b` qui prend en entrée un nom de fichier CSV du format donné ci-dessus et qui renvoie une liste d'association (dictionnaire) dont les clefs sont des paires (Station,Réseau) et les valeurs leur fréquentation. La valeur doit être un entier.

12 Écrire une fonction `question11b` qui prend en entrée un nom de fichier CSV du format donné ci-dessus et qui renvoie un dictionnaire dont les clefs sont les lignes de Métro avec comme valeur la station la moins fréquentée sur cette ligne.