

TD n° 3

1 Jeu de mots

Le but du jeu est de partir d'un mot de peu de lettres (par exemple "ode"), et de proposer une suite de mots qui sont construits à partir des lettres du précédent, en ajoutant une lettre (avec bien sûr un sens en français).

Par exemple, à partir de "ode", on peut proposer "dose", puis "codes", etc.

Exercice 1. Inclusion

On dira qu'un mot **mot1** *est inclus dans* un autre mot **mot2** si chaque lettre de **mot1** apparaît dans **mot2**, mais n'apparaît pas plus fréquemment dans **mot1** que dans **mot2**.

Par exemple,

- "abc" est inclus dans "cadb",
- "menee" n'est pas inclus dans "monnaie" (car le "e" apparaît plus fréquemment dans "menee" que dans "monnaie").

Écrivez une fonction *est_inclus_dans* qui prend en arguments deux chaînes de caractères **mot1** et **mot2** et qui retourne un booléen qui dit si oui ou non **mot1** est inclus dans **mot2**.

Vous pourrez, dans votre réflexion, vous demander à quelle condition cette fonction renvoie **False**. Vous pourrez aussi utiliser la méthode **count** des chaînes de caractères (diapo 35 du CM 02).

Exercice 2. Longueurs correctes

Écrivez une fonction *longueurs_correctes* qui prend en arguments deux chaînes de caractères **mot1** et **mot2** et qui retourne un booléen qui dit si oui ou non la longueur de **mot2** est égale à celle de **mot1** plus un.

Exercice 3. Successeur

Écrivez une fonction *est_un_successeur* qui prend en arguments deux chaînes de caractères **mot1** et **mot2** et qui retourne un booléen qui dit si oui ou non **mot2** est un successeur possible de **mot1** dans le cadre du jeu.

Exercice 4. Une fonction de présentation des mots

Écrivez une fonction *afficher* qui prend en argument une liste de chaînes de caractères, et qui les affiche en allant à la ligne après chaque chaîne (ainsi, cet affichage permettra de mieux voir l'évolution du jeu coup après coup).

Exercice 5. Programme du jeu

Écrivez le programme principal qui :

- demande le mot initial à l'utilisateur ;
- puis, tant que le jeu n'est pas fini, demande sa proposition de nouveau mot à l'utilisateur. Le jeu s'arrête si la proposition n'est pas un successeur du mot précédent, ou si l'utilisateur entre "0" comme proposition ;
- affiche à chaque tour l'état actuel de la liste des mots proposés (et valides) ;
- à la fin, affiche soit "jeu interrompu par l'utilisateur", soit "jeu interrompu par erreur de mot".

Voici deux captures de situations :

```
Quel est le mot initial ? pot
pot
entrez votre proposition : trop
pot
trop
entrez votre proposition : porte
pot
trop
porte
entrez votre proposition : porter
pot
trop
porte
porter
entrez votre proposition : 0
jeu interrompu par l'utilisateur
```

```
Quel est le mot initial ? mur
mur
entrez votre proposition : mure
mur
mure
entrez votre proposition : brome
jeu interrompu par erreur de mot
```

2 Flotte de bateaux

On représente un bateau par un tuple (*type_bateau*, *nom_bateau*, *vitesse*, *distance_au_port*).

Par exemple, le bateau représenté par le tuple ("**voilier**", "**Albatros**", **12**, **36**).

Le voilier "Albatros" navigue à 12 noeuds et est localisé à 36 milles nautiques de son port d'attache.

On représente aussi une flotte par une liste de bateaux.

Par exemple [(**"voilier"**, **"Albatros"**, **12**, **36**),(**"chalutier"**, **"Eole"**, **2**, **0**),(**"ferry"**, **"Sirius"**, **8**, **32**)] représente une flotte de 3 bateaux.

On considère que le nom d'un bateau est unique, il sert donc d'identifiant.

Un bateau peut être :

- "amarré au port" si sa distance au port est égale à 0 ;
- "en manoeuvre dans le port", si sa distance au port est inférieure à 1 et si sa vitesse n'est pas nulle ;
- "au mouillage en mer" si sa distance au port est supérieure ou égale à 1 et sa vitesse est nulle ;
- "en manoeuvre en mer" si sa distance au port est supérieure ou égale à 1 et sa vitesse est non nulle.

Exercice 1. Bateaux d'un type donné

Écrivez une fonction *bateaux_de_type* qui prend en paramètres une flotte et une chaîne de caractères, et qui retourne la liste des noms des bateaux de la flotte qui sont du type précisé.

Exercice 2. Bateaux en mer

Écrivez une fonction *bateaux_en_mer* qui prend en argument une flotte, et qui retourne la liste des noms des bateaux de cette flotte qui sont en mer (au mouillage ou en manoeuvre).

Exercice 3. Le plus rapide

Écrivez une fonction *le_plus_rapide* qui prend en argument une flotte, et qui retourne le nom du bateau le plus rapide de la flotte.

Exercice 4. Retour au port

Écrivez une fonction *retour_au_port* qui prend en argument une flotte, et qui retourne une liste de tuples de la forme (*nom_bateau*, *temps_pour_arriver*). Ces tuples représentent les bateaux de la flotte qui étaient en mer (mais pas au mouillage), et à qui on a ordonné de retourner immédiatement au port.

Le temps pour arriver est donc à calculer en fonction de leur vitesse et de leur distance au port. La liste à retourner n'a pas à être triée.

3 Prendre en compte les marées pour ne pas s'échouer ...

Exercice 1.

Un navigateur a besoin de savoir s'il pourra passer en sécurité à un endroit donné, ou s'il risquera de s'échouer par manque de profondeur d'eau !

L'annuaire des marées permet au marin de connaître les hauteurs d'eau à marée haute et à marée basse du lieu qui l'intéresse. Le marin peut repérer la dernière marée qui aura lieu à une certaine heure H, avant son passage. Il sait s'il s'agit d'une marée haute ou d'une marée basse.

Connaissant ainsi le sens de la prochaine marée (montante ou descendante), il souhaite calculer la hauteur d'eau x heures après (donc à $H+1$, $H+2$,... ou $H+6$).

On notera que si la mer était basse à l'heure H , elle est pleine à l'heure $H+6$ (et inversement), l'inversion se faisant toutes les 6 heures.

Par exemple :

- si à 15h a lieu une marée haute, et si le marin souhaite passer à 17h, il passera donc à $H+2$, en marée descendante.
- si à 11h a lieu une marée basse, et si le marin souhaite passer à 16h, il passera donc à $H+5$ en marée montante.

Pour effectuer le calcul de la hauteur d'eau à $H+x$, on commence par déterminer le marnage (c'est-à-dire la différence entre la hauteur d'eau de la pleine mer et celle de la basse mer), ainsi que la hauteur d'eau correspondant à un douzième du marnage.

Par exemple, si la marée haute est à 8.80 m et la marée basse à 3.50 m, alors le marnage vaut $8.80 - 3.50 = 5.30$ m, et le douzième du marnage vaut $5.30 / 12 = 0.44$ m.

La montée (ou la baisse) des eaux respecte (en simplifiant) les règles suivantes :

- durant la première heure la mer monte (ou baisse) de 1/12e (soit de 0.44 m avec notre exemple),
- durant la deuxième heure la mer monte (ou baisse) de 2/12e (soit de 0.88 m),
- durant la troisième heure la mer monte (ou baisse) de 3/12e (soit de 1.32 m),
- durant la quatrième heure la mer monte (ou baisse) de 3/12e (soit de 1.32 m),
- durant la cinquième heure la mer monte (ou baisse) de 2/12e (soit de 0.88 m),
- durant la sixième heure la mer monte (ou baisse) de 1/12e (soit de 0.44 m).

On peut calculer de combien de mètres la mer aura monté (ou baissé) dans un délai compris entre 1 et 6 heures.

Avec notre exemple, après la 2ème heure de marée, la mer aura monté (ou baissé) de $0.44 + 0.88$ soit de 1.32 mètre. Pour connaître la hauteur d'eau demandée, il suffira d'ajouter (ou retrancher) ce chiffre à la hauteur de la marée de l'heure H , selon le sens de la marée (montante ou descendante).

Écrire une fonction *hauteur_maree(basse_mer, haute_mer, d, sens_montant)* prenant en arguments les hauteurs d'eau à marée basse et à marée haute, un entier d correspondant au délai (en heures) souhaité pour le calcul, ainsi qu'un booléen *sens_montant* (=True si la prochaine marée sera montante, False sinon). La fonction retournera la hauteur d'eau totale à l'heure $H+d$.