Le mot le plus long

Dans ce TP, vous allez programmer une fonction qui permet de trouver le mot le plus long que l'on puisse réaliser avec un ensemble de lettres donné.

Exercice 1

Dans cette première partie, vous allez écrire quelques fonctions de manipulation de listes. Ces fonctions serviront dans la seconde partie.

1. Écrire une fonction `longueur` qui calcule la longueur d'une liste.
2. Écrire une fonction `est_dans` qui prend un élément x et une liste l et teste si x apparaît dans l .
3. Écrire une fonction `enleve` qui prend un élément x et une liste l et renvoie la liste l où la première occurrence de x a été enlevée. Renvoie une erreur explicite si x n'apparaît pas dans l .
4. Écrire la fonction `ajout_fin` qui prend un élément x , une liste l et renvoie la liste où x a été ajouté à la fin de l .
5. En utilisant cette fonction, écrire une fonction `retourne` qui prend une liste l et renvoie la liste comportant les mêmes éléments que l , mais en ordre inverse.
Ainsi, `retourne [1;2;3] = [3;2;1]`

Exercice 2

Le but de cette partie est de vous faire écrire une fonction permettant de trouver les mots les plus longs pouvant être formés à partir des lettres d'un premier mot. Télécharger, dans le même répertoire que votre TP, les fichiers `dico-francais.txt` et `mot.ml`, puis exécutez la commande `#use "mot.ml";;`.

Ce fichier vous donne accès à un type de données `string` représentant les mots, une fonction `lettres_d_un_mot` — qui prend un mot et renvoie la liste des lettres formant ce mot — et une variable `dico` qui contient la liste des mots du dictionnaire. Pour écrire un mot, de type `string` on utilisera les guillemets.

Exemple : `lettres_d_un_mot "maison" = ['m';'a';'i';'s';'o';'n']`.

Ainsi, la fonction `longueur_mot`, qui calcule la longueur d'un mot, s'écrit :

```
let longueur_mot m = longueur (lettres_d_un_mot m)
```

1. Écrire une fonction `est_sous_ensemble` qui prend en arguments deux listes de lettres et qui renvoie un booléen indiquant si l_1 est un sous ensemble de l_2 .

Par exemple :

```
est_sous_ensemble ['a';'b'] ['b';'c';'a'] = true
est_sous_ensemble ['a';'b';'c'] ['b';'c';'a'] = true
est_sous_ensemble ['a';'b';'c';'d'] ['b';'c';'a'] = false
est_sous_ensemble ['a';'a';'b'] ['b';'c';'a'] = false
est_sous_ensemble ['a';'a';'b'] ['b';'a';'c';'a'] = true
```

On utilisera les fonctions `est_dans` et `enleve` de la partie 1.

2. En déduire une fonction `peut_s_ecrire_avec` qui prend en arguments deux chaînes de caractères et qui renvoie un booléen indiquant si l'on peut écrire `m1` avec des lettres issues de `m2`.
3. Utiliser cette fonction pour écrire une fonction `selectionne_peut_s_ecrire_avec` qui à un mot `m` et une liste de mots `l` associe la liste des mots de `l` pouvant s'écrire avec les lettres figurant dans `m`.
Le prototype de cette fonction devra être : `string -> string list -> string list`
4. Écrire une fonction `max` qui prend deux entiers et renvoie le plus grand. En déduire une fonction `longueur_plus_long` qui prend une liste de mots et renvoie la longueur du mot le plus long. On utilisera la fonction `longueur_mot`.
5. Écrire une fonction `cherche_mots_longueur` qui prend un entier `n` et une liste de mots `l` et renvoie la liste des mots de longueur `n` figurant dans `l`. Sachant que "anticonstitutionnellement" est le mot le plus long du dictionnaire, testez les fonctions `longueur_plus_long` et `cherche_mots_longueur` à l'aide de la liste `dico` des mots du dictionnaire fournie.
6. Enfin, écrire la fonction `les_mots_les_plus_long` qui prend un mot `m` et renvoie les mots de longueur maximal que l'on peut former avec les lettres du mot `m`. On utilisera les fonctions réalisées dans les questions précédentes (3,4 et 5) et la liste `dico` des mots du dictionnaire fournie.