TD n°9 - Chaines de caractères en Ocaml

Le but de ce TD est la manipulation de chaines de caractères en Ocaml.

Les chaines de caractères sont immuables mais permettent l'accès à chacun de leur caractères via son indices.

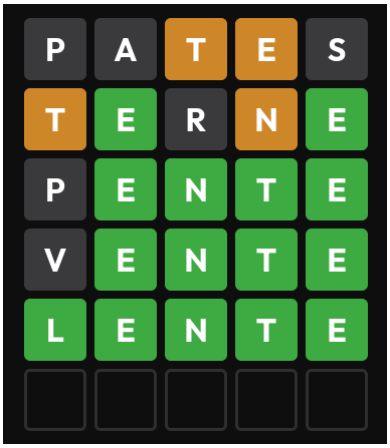
Quelques utilitaires :

- Pour accéder au caractère `i` de `s` on écrit `s.[i]`.
- Pour créer une chaine de caractère de taille `n` dont tous les caractères sont `c`, on écrit `String.make n c`.
- Pour concaténer des chaines de caractères, on utilise `^`.

1 Jouons à Wordle

Wordle est un jeu dans lequel le joueur doit deviner un mot de 5 lettres en proposant des mots de 5 lettres.

Pour chaque mot proposé, l'interface indique si les lettres sont présentes dans le mot (en orange), présentes dans le mot et à cette position (en vert) ou non présentes dans le mot (en gris).



Le joueur possède un nombre fixé de tentatives et gagne s'il propose le bon mot pendant ses tentatives.

On va implémenter ce jeu en Ocaml en généralisant : la taille des mots à deviner/proposer sera notée n et le nombre de tentatives sera noté m.

On représentera le mot à deviner et les essais du joueur par des chaines de caractères.

Pour gérer les couleurs, on utilisera des tableaux de la même longueur que la chaine de caractère qui contiennent des entiers :

- 0 si c'est gris
- 1 si c'est orange
- 2 si c'est vert

1. Écrire une fonction `est_dans : string -> char -> bool` qui prend en entrée une chaine de caractères s et un caractère c et renvoie `true` si c est dans s et `false` sinon.
2. Écrire une fonction `couleurs : string -> string -> int array` qui prend en entrée le mot à deviner et une proposition et donne la couleur de chacune des lettres.
   Pour simplifier, on ne s'inquiétera pas des possibles doublons de lettres.

On suppose écrite une fonction `affiche_couleurs : string -> int array -> unit` qui prend en entrée une proposition et le tableau de couleurs associés et affiche les lettres en vert, en orange ou en blanc (sur fond noir) selon si la lettre est à sa place, est dans le mot ou n'est pas dans le mot.

3. Écrire une fonction `un_tour : string -> bool` qui prend en entrée le mot à deviner s et lit une proposition, affiche la proposition avec les couleurs et renvoie `true` si le joueur a gagné et `false` sinon.
4. Écrire une fonction `wordle : int -> string -> unit` qui prend en entrée un entier m et le mot à deviner s, et fait jouer l'utilisateur au jeu, en le limitant à m propositions. La fonction affiche à la fin si le joueur a gagné ou pas.
5. Écrire une fonction `couleurs_bis : string -> string -> int array` qui prend en entrée le mot à deviner et une proposition et donne la couleur de chacune des lettres.

Cette fois-ci on va prendre en compte les doublons, c'est à dire que s'il y a deux e dans la proposition et un e dans le mot, alors on devrait afficher un e en orange/vert et un e en gris, avec une priorité pour le e vert sil existe.

Ce qu'on peut faire est d'abord vérifier les lettres vertes. Ensuite, pour chaque lettre c apparaissant dans la proposition, on compte combien de fois elle apparait dans la solution, on note `nb` ce nombre. On compte ensuite combien de lettre c de la proposition sont à la bonne place (vertes), on note `nbv` ce nombre. Le nombre de c à mettre en orange dans l'affichage est alors `nb - nbv`.

Par exemple si `s="lalala"` et que la proposition est `"aaaaaa"`, alors on mettra en vert les 3 'a' à la bonne place et on ignorera les autres car on a 3 'a' au total - 3 'a' verts.

Si la proposition est `"aaabab"`, alors on a 3 'a' au total - 1 'a' vert. On doit donc indiquer 2 'a' oranges et le 'a' restant est indiqué comme gris, car on a déjà assez de 'a'.