TP D'OPTION INFORMATIQUE 1
Expressions rationnelles

On utilise pour implémenter en OCaml les expressions rationnelles (standards, non vides) le type :

```
type 'a regexp =
    Epsilon |
    Lettre of 'a |
    Plus of 'a regexp * 'a regexp |
    Concat of 'a regexp * 'a regexp |
    Etoile of 'a regexp ;;
```

où le paramètre de type `'a` correspond au type des éléments de Σ (usuellement `char`).

1. Représenter l'expression $(a+b)^*ca$ dans le type `char regexp`.
2. Écrire une fonction `taille : 'a regexp -> int` prenant en argument une expression rationnelle et renvoyant sa taille, ie le nombre de constructeurs y apparaissant.
3. On veut à présent calculer les ensembles de 1-préfixes, 1-suffixes, 2-facteurs d'une expression rationnelle. L'algorithme vu en cours nécessite de pouvoir calculer des unions et des concaténations d'ensembles de lettres. Les ensembles seront représentés par des listes **triées sans doublons**.
   (a) Écrire une fonction `union : 'a list -> 'a list -> 'a list` prenant en argument deux ensembles, et renvoyant leur union.
   (b) Écrire une fonction `concat : 'a list -> 'b list -> ('a * 'b) list` prenant en argument deux ensembles de lettres, et renvoyant leur produit cartésien, sous forme d'ensemble de couples de lettres.
   Par exemple, `concat ['a'; 'b'] ['c'; 'd'; 'e']` doit renvoyer
   `[('a', 'c'); ('a', 'd'); ('a', 'e'); ('b', 'c'); ('b', 'd'); ('b', 'e')]`
   (c) Écrire une fonction `psf : 'a regexp -> bool * 'a list * 'a list * ('a * 'a) list` prenant en argument une expression rationnelle supposée linéaire et renvoyant le quadruplet (b, P, S, F) caractérisant le langage local dénoté par cette expression.
   Par exemple, le quadruplet renvoyé sur l'expression $(a+b)^*c$ doit être

   ```
   false,
   ['a'; 'b'; 'c'],
   ['c'],
   [('a', 'a'); ('a', 'b'); ('a', 'c'); ('b', 'a'); ('b', 'b'); ('b', 'c')]
   ```

   (d) Rappeler la complexité de la fonction précédente.
4. Écrire une fonction
   `reconnu_local : bool * char list * char list * (char * char) list -> string -> bool`
   prenant en argument un quadruplet (b, P, S, F) caractérisant un langage local, et un mot m, sous forme de chaîne de caractères, et décidant si m appartient à ce langage local. Quelle est la complexité de cette fonction relativement à la longueur de m ?
5. Écrire une fonction `est_lineaire : 'a regexp -> bool` prenant en argument une expression rationnelle et déterminant si elle est linéaire. On demande une complexité linéaire en la taille de l'expression. On pourra utiliser les fonctions suivantes, en $O(1)$:
   - `Hashtbl.create 0` : crée un dictionnaire vide ;
   - `Hashtbl.mem h a` : teste si la clé `a` apparait dans le dictionnaire `h` ;
   - `Hashtbl.add h a b` : ajoute la valeur `b` associée à la clé `a` dans le dictionnaire `h`.
6. Ecrire une fonction `enumerer : 'a regexp -> int -> 'a list list` prenant en argument une expression rationnelle et un entier naturel n, et renvoyant l'ensemble des mots d'au plus n lettres reconnus par cette expression.