AUTOMATES

Vous trouverez sur le site du lycée des automates déterministes et non-déterministes prêts à l'emploi, ainsi que quelques petites fonctions utiles.

L'alphabet étant connu, un automate (déterministe ou non) est défini par :
- ses états ;
- son ou ses états initiaux ;
- ses états finaux ;
- ses transitions, qui sont des triplets (p, a, q) où p, q sont deux états et a une lettre de l'alphabet.

Autrement dit, avec cette définition, un automate est un quadruplet de 4 ensembles.

Pour représenter l'alphabet, on définit un type composé de constantes. Pour nous, ce sera un alphabet à deux lettres. Mais on fera comme si l'alphabet pouvait être quelconque.

```
type lettre = a | b;;
let alphabet = [a; b];;
```

Les mots seront alors des listes de lettres.

On peut facilement représenter un automate en CAML par un enregistrement, dont les champs sont des listes. Pour permettre avec le même type de manipuler aussi bien les automates déterministes ou les non-déterministes, on donne un paramètre de type, qui précise la nature des états :

```
type 'a automate = {
  etats   : 'a list;
  initial : 'a list;
  final   : 'a list;
  trans   : ('a * lettre * 'a) list;
```

Pour distinguer un automate déterministe d'un non-déterministe, il suffit de vérifier que le champ initial est une liste à un élément et que dans le champ trans, il n'y a pas deux triplets (p, a, q) et (p', a', q') tels que $p = p'$ et $a = a'$.

1) Écrivez une fonction delta auto e x de type `'a automate -> 'a -> lettre -> 'a list` qui calcule la liste des états accessibles dans l'automate auto depuis l'état e en suivant une transition étiquetée par la lettre x.

Si l'automate est déterministe, les listes ainsi calculées ne peuvent donc être que vides (blocages) ou à un seul élément.

Si l'automate est non-déterministe, les listes ainsi calculées peuvent être quelconques.

2) Si on suppose l'automate déterministe, écrivez une fonction reconnu mot auto de type `lettre list -> 'a automate -> bool`, qui détermine si le mot est reconnu ou pas par l'automate.

La suite est consacrée à l'écriture d'un algorithme de déterminisation d'un automate.

3) Écrivez une fonction Delta auto le x de type `'a automate -> 'a list -> lettre -> 'a list`, qui calcule la liste des états accessibles dans l'automate auto depuis n'importe quel état d'une liste d'états le en suivant une transition étiquetée par la lettre x. C'est ici qu'il est important de ne pas répéter le même état dans la liste : les listes représentent des ensembles !

4) Écrivez une fonction calcul_transitions auto le de type `'a automate -> 'a list -> ('a list * lettre * 'a list) list`, qui calcule la liste de tous les triplets (le, x, Delta auto le x), c'est-à-dire les transitions d'un état de l'automate déterminisé vers tous ses états suivants.

Dans la suite, ouv et fer sont deux listes de listes d'états (vous aurez compris que ce sont les ensembles ouvert et fermé de l'algorithme de déterminisation).

5) Écrivez une fonction ajouter ouv fer lt, de paramètres une liste ouverte, une liste fermée et une liste de transitions (le, x, le') (comme celles calculées ci-dessus) et qui calcule une nouvelle liste ouverte en ajoutant à celle-ci les listes d'états le' quand elles ne sont pas déjà dans ouv ou fer.

6) Pour conclure, écrivez une fonction determ auto de type `'a automate -> 'a list automate`, qui calcule l'automate déterminisé équivalent à un automate quelconque.