info421 : Programmation fonctionnelle
TD 3 : programmes récursifs

Responsable : Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
email : Pierre.Hyvernat@univ-savoie.fr
http://lama.univ-savoie.fr/~hyvernat/

Exercice 1 : dictionnaires

Il est souvent utile d'avoir un "dictionnaire" : une structure de données qui permet d'associer une valeur à une clé. C'est une espèce de tableau généralisé, où les indices ne sont pas forcement des entiers consécutifs. Une maniere de voir un dictionnaire, c'est comme un ensemble de *cases*. Chaque case a un nom (une clé) et une valeur.

> **Remarque :** il y a au plus une case avec une clé donnée ; par contre, il peut y avoir plusieurs cases avec la meme valeur...

La manière la plus simple (et la plus naïve) est de définir un dictionnaire comme une liste de couples : si la valeur `42` est associés à la clé `"Answer"`, alors la liste représentant le dictionnaire contiendra la couple `("Answer" , 42)`.

Question 1. Définissez le type `('a,'b) dict` des dictionnaires dont les clés sont de type `'a` et les valeurs de type `'b`.

Question 2. Définissez les fonctions
- : `insere : 'a -> 'b -> ('a,'b) dict -> ('a,'b) dict`
- : `supprime : 'a -> ('a,'b) dict -> ('a,'b) dict`
- : `existe_cle : 'a -> ('a,'b) dict -> bool`
- : `existe_val : 'b -> ('a,'b) dict -> bool`
- : `recherche_val : 'b -> ('a,'b) dict -> 'a list`
- : `recherche_cle : 'a -> ('a,'b) dict -> 'b option`

Question 3. Écrivez deux fonctions pour obtenir la liste des clés et la liste des valeurs présentes dans un dictionnaire. Quel est le type de ces fonctions ?

Question 4. Écrivez une fonction `zip` qui permet de transformer une liste de clés et une liste de valeurs en un dictionnaire correspondant.

Que faire si les deux listes n'ont pas la même taille ?

Question 5. Comment pourrait-on optimiser cette structure de données ?

Exercice 2 : listes, définitions récursives

Question 1. Écrivez une fonction qui permet de supprimer les doublons dans une liste.

Question 2. On peut représenter un ensemble par la liste de ses valeurs. Par exemple, l'ensemble $\{0, 1, 4\}$ peut être représenté par la liste `[0 ;1 ;4]`.

Il y a exactement 2^n sous-ensembles d'un ensemble à n élements. Si x est un élément de X, les sous-ensembles de X sont séparés en deux parties :
- ceux qui ne contiennent pas x, que l'on peut obtenir récursivement,
- ceux qui contiennent x, que l'on peut obtenir en ajoutant x aux précédents.

Bien entendu, il y a un unique sous-ensemble de l'ensemble vide : l'ensemble vide lui même.

On veut écrire une fonction `sousEnsemble` qui calcule la liste des sous-ensembles d'un ensemble (vu comme une liste).

Quel est le type de cette fonction ?

Transformez ces remarques en une définition récursive, et programmez la fonction `sousEnsemble`.

Par exemple :

```
sousEnsemble [1;2;4] ;;
- : int list list = [[]; [4]; [2]; [2; 4]; [1]; [1; 4]; [1; 2]; [1; 2; 4]]
```

Question 3. Étant donnée une liste, on aimerait générer tous les mélanges possibles de cette liste. Par exemple

```
melanges [1;2;4] ;
- : int list list = [[1; 2; 4]; [2; 1; 4]; [2; 4; 1];
                     [1; 4; 2]; [4; 1; 2]; [4; 2; 1]]
```

Si la liste est vide, il n'y a qu'un seul mélange possible : la liste vide elle même.

Si la liste contient l'élément `a`, on peut procéder récursivement :

- en génère tous les mélanges possible pour la liste *sans l'élément* `a`,
- pour chaque mélange obtenu, on génère toutes les possibilités d'insérer `a` dans le mélange,
- on garde tous les possibilités pour tous les mélanges.

Pour programmer ceci :

1- donnez le type de la fonction `insere` qui prend une liste et un élément, et renvoie la liste de toutes les manières possibles d'insérer l'élement dans la liste.

2- programmez la fonction `insere`

Si on utilise la fonction `List.map` pour appliquer la fonction `insere a` sur la liste des mélanges obtenus récursivement, on obtient une liste de liste de mélanges :

```
- : int list list list = [[[1; 2; 3]; [2; 1; 3]; [2; 3; 1]];
                          [[1; 3; 2]; [3; 1; 2]; [3; 2; 1]]]
```

Programmez une fonction `union` qui permet de mettre toutes ces listes bout à bout pour obtenir simplement une liste de mélanges.

```
- : int list list = [[1; 2; 3]; [2; 1; 3]; [2; 3; 1];
                     [1; 3; 2]; [3; 1; 2]; [3; 2; 1]]
```

Quelle est le type de la fonction `union` ?

Écrivez la fonction `melanges`.