info421 : Programmation fonctionnelle
TD 4 : programmes récursifs

Responsables : Pierre Hyvernat et Krzysztof Worytkiewicz
Laboratoire de mathématiques de l'université de Savoie
email : Pierre.Hyvernat@... / Krzysztof.Worytkiewicz@... (...@univ-savoie.fr)
http://lama.univ-savoie.fr/~hyvernat/
http://lama.univ-savoie.fr/~worytkiewicz/

Exercice 1 : petits exercices combinatoires

Ces questions sont volontairement très concises. Essayez de décomposer le problème avant de vous lancer dans une solution...

Question 1. Comment générer la liste de tous les "sous-ensembles" de $\{0, ..., n-1\}$?

Par exemple, pour l'entier `3`, vous devez générer la liste (pas forcement dans cet ordre)

```
[ [] ; [0] ; [1] ; [2] ; [0 ;1] ; [1 ;2] ; [0 ;2] ; [0 ;1 ;2] ]
```

Question 2. Comment générer la liste de toutes les permutations sur l'ensemble $\{0, ..., n-1\}$?

Par exemple, pour l'entier `3`, vous devez générer la liste (pas forcement dans cet ordre)

```
[ [0 ;1 ;2] ; [1 ;0 ;2] ; [1 ;2 ;0] ; [0 ;2 ;1] ; [2 ;0 ;1] ; [2 ;1 ;0] ]
```

Question 3. Comment faire pour supprimer les doublons dans une liste ?

Exercice 2 : dictionnaires

Il est souvent utile d'avoir un "dictionnaire"* : une structure de données où l'on associe 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_cle : 'a -> ('a,'b) dict -> 'b option
- : recherche_val : 'b -> ('a,'b) dict -> 'a list
```

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 ?

* autres noms : *hash table*, *liste d'associations*, ...

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

Exercice 3 : arbres binaires

Question 1. En reprenant le type des arbres du TD3, programmez la fonction `profondeur` d'un arbre.

Question 2. Comment vérifier si un arbre est *équilibré* : à `1` près, toutes les branches ont la même hauteur.

Question 3. Programmez la fonction `present : 'a -> 'a arbre -> bool` qui vérifie si un élément est présent dans un arbre.

On considère maintenant des arbres binaires particuliers : les *arbres binaires de recherche*. Ce sont des arbres binaires qui ont la propriété que :
- les éléments du sous-arbre gauche sont plus petit que l'élément d'un noeud,
- les éléments du sous-arbre droit sont plus grand que l'élément d'un noeud.

Question 4. Reprogrammez une fonction `present_bst` qui vérifie si un élément est présent dans un arbre binaire de recherche.

Question 5. Programmez une fonction `insere : 'a -> 'a arbre -> 'a arbre` qui insère un élément dans un arbre binaire de recherche, tout en maintenant la propriété.

Question 6. Programmez une fonction qui vérifie si un arbre est *binaire de recherche*.