info401 : Programmation fonctionnelle
TD 3 : types inductifs

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22, poste : 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/
wiki : http://www.lama.univ-savoie.fr/wiki

Exercice 1 : les arbres binaires

On considère les arbres binaires avec des données sur les noeuds. Autrement dit, un arbre étiqueté par des entiers est :
- soit l'arbre vide
- soit un noeud étiqueté (par une donnée) avec deux fils : le fils gauche (un arbre) et le fils droit (un arbre).

Question 1. Donnez le type inductif correspondant.

Question 2. Écrivez la fonction `taille` de type `'a arbre -> int` qui calcule combien de noeuds possède un arbre.

Question 3. Écrivez un équivalent de la fonction `List.map` qui applique une fonction à tous les noeuds d'un arbre. Son type sera `('a -> 'b) -> 'a arbre -> 'b arbre`.

Question 4. Écrivez la fonction qui prend en argument une fonction de poids, et calcule la somme des poids des noeuds d'un arbre : `poids : ('a -> int) -> 'a arbre -> int`

Question 5. À votre avis, que serait l'équivalent de la fonction `List.fold_right` (`reduit`) pour le type des arbres.

Programmez cette fonction, et servez vous en pour redonner une définition de `taille`, `applique` et `poids`.

Question 6. Utilisez la fonction `fold` pour programmer la fonction qui collecte toutes les données d'un arbre dans une liste.

Question 7. Même question que précédemment, mais si on veut seulement collecter les feuilles de l'arbre.

Exercice 2 : la tortue

Tartampion a décidé de réécrire une partie du langage LOGO en Caml. La partie qui l'intéresse est la partie "graphique" : la *tortue* est un point qui possède trois caractéristiques :
- une position (coordonnées dans le plan),
- une orientation (angle par rapport à la verticale),
- un crayon (booléen : `false` pour dire que son crayon est levé).

Nous nous intéressons seulement aux opérations que la tortue peux effectuer :
- `forward` pour avancer d'une certaine distance,
- `left` pour tourner vers sa gauche d'un certain angle en degrés,
- `setpos` pour "téléporter" la tortue à une nouvelle position (coordonnées)
- `setheading` pour fixer l'orientation de la tortue par rapport à la verticale,
- `penup` et `pendown` pour lever ou baisser le crayon.

Question 1. Donnez un type de données pour représenter les différentes actions de la tortue.

Question 2. Écrivez une fonction qui prend en argument un liste d'actions de la tortue et qui renvoie `true` si la tortue écrit quelque chose et `false` sinon.

Par exemple, sur `[ setpos (0.,0.) ; left 90. ; penup ; forward 100. ]`, votre fonction devra répondre `false`.

Question 3. Écrivez (succinctement) une fonction qui aura pour arguments :
- une position initiale (coordonnées),
- une orientation initiale (angle),
- une liste d'action,

et qui calculera la position et l'orientation de la tortue après les actions de la liste.

Question 4. Réfléchissez à la remarque suivante : certaines opérations peuvent se regrouper : `[ forward 100. ; forward 50. ]` est équivalent à `[ forward 150. ]`. Comment simplifier au maximum une liste de commandes ?

Que dire du cas où on rajoute des instructions `backward` et `right`, avec des interprétations évidentes ?

Question 5. On veut maintenant rajouter la possibilité de répéter des actions un certain nombre de fois. Par exemple `repeat 4 [ forward 10. ; left 90. ]` permettra de dessiner un carré.

Donnez une nouvelle définition du type des actions de la tortue pour prendre ceci en compte.