Fiche de TD-TP no. 5

Évaluation

Exercice 1. Utilisez d'abord la stratégie *call-by-value* et ensuite la stratégie *call-by-name* pour évaluer les expressions suivantes :

```
f (True && True) [] (replicate 2 '*') where f b x y = if b then x else y
take 2 (repeat '*')
sum (Succ Zero) Zero
```

Exercice 2. Rappelons les definition des fonctions `foldr`, `foldl`, `map`, et `(++)` :

```
foldr :: (a -> b -> b) ->  b ->  [a] -> b
foldr f v [] = v
foldr f v (x:xs) = f x (foldr f v xs)

foldl :: (a -> b -> a) -> a -> [b] -> a
foldl f v [] = v
foldl f v (x:xs) = foldl f (f v x) xs

map :: (a -> b) -> [a] -> [b]
map f [] = []
map f (x:xs) = f x:map f xs

(++) :: [a] -> [a] -> [a]
[] ++ ys = ys
(x:xs) ++ ys = x:(xs ++ ys)
```

Evaluez, par nom et par valeur, les deux expressions suivantes :

```
exp1 = foldr (++) [] (map (\ x -> [x]) [1,2])
exp2 = foldl (++)  [] (map (\ x -> [x]) [1,2])
```

Raisonner sur les programmes

Exercice 3. Démontrez les égalités suivantes :

```
[] ++ ys == ys
xs ++ [] == xs
(xs ++ ys) ++ zs == xs ++ (ys ++ zs)
```

Exercice 4. Rappelez les définitions des fonctions `length` et `replicate` et, ensuite, démontrez l'égalité suivante :

```
length (replicate x n) == n
```

Types recursifs

Nous allons considérer, dans la suite, le type des arbres binaires et les fonctions `estFeuille`, , `foldArbre` définis comme suit :

```
data Arbre = Feuille Int | Noeud Arbre Int Arbre

estFeuille (Feuille _) = True
estFeuille _ = False

taille x = case x of
             Feuille _ -> 1
             Noeud l _ r -> taille l + taille r + 1

foldArbre f g v (Feuille n) = g n v
foldArbre f g v (Noeud l n r ) = f (foldArbre f g v l) n (foldArbre f g v r)
```

Exercice 5. Repondez aux questions suivantes :

1. Parmi les fonctions définies ci-dessus, lesquelles utilisent le filtrage dans leur définition, et lesquelles utilisent la récursion ?
2. Quel est le type de ces fonctions ?
3. Définissez une fonction

   ```
   label : Arbre -> Int
   ```

   qui (de façon analogue a la fonction `head` pour les listes) retourne l'étiquette d'un arbre de la forme `Noeud l n r`.
4. Proposez une définition récursive d'une fonction `somme` qui calcule la somme de toutes les étiquettes.
5. Proposez une définition de la même fonction, via `foldArbre`.

En TP

Discuter et implémenter le projet.