Fiche de TD no. 1

Listes et fonctions sur les listes

Exercice 1 : *Expressions de liste.* Une liste s'écrit entre crochets, avec les éléments de la liste séparés par des virgules.

1. Rappelez ce que font les opérateurs `[],:,++,..` et les fonctions `head`, `tail`, `reverse`.
2. Évaluez les expressions de liste suivantes :
   (a) `1:[2]`
   (b) `[3,4]++[1,2]`
   (c) `[3..10]`
   (d) `tail [1..4] ++ 5:[]`
   (e) `head [1..4] : [5]`
   (f) `reverse [1..4] ++ [5]`

Exercice 2 : *Définition de fonctions sur les listes.* Rappelez ce que font les fonctions `head`, `tail`, `reverse`, `length`, `drop`, `take`, `!!`, définie dans `prelude.hs` et vue en cours.

1. La fonction `last`, définie dans `prelude.hs`, sélectionne le dernier élément d'une liste. Montrez comment cette fonction peut se définir en utilisant les fonctions sur le listes présentées dans le cours.
2. Pouvez vous penser à d'autres définitions de `last` ?
3. De façon semblable, montrez comment la fonction `init`—du `prelude.hs`, qui enlève le dernier élément d'une liste—-peut se définir de plusieurs façons.

Exercice 3 : *Fonctions sur les listes.* Une chaîne de caractères est une liste de caractères. On souhaite définir une fonction appelée `palindrome` qui decide si une chaîne de caractères est un palindrome (on peut lire cette chaîne du début ou de la fin, en obtenant le même résultat).

1. Quel est le type de cette fonction ? Après avoir explicité le domaine (le type des arguments) et le codomaine (le type des valeurs retournés) de la foction, écrivez l'expression de type appropriée.
2. Proposez une définition de cette fonction.

Application

Exercice 4 : *Maths et Haskell.*

1. Considérez les expressions `Haskell` suivantes :
   (a) `f 3 + g 9 * 8` ,
   (b) `f 22 33 * (g 44 h 55)` ,
   (c) `f 22 33 * g 44 (h 55)` .
   — Écrivez un petit trait la où il se trouve l'opérateur d'application.
   — Écrivez l'arbre syntaxique de la deuxième et la troisième expression.
   — Réécrivez ces expressions en notation mathématique.
   — Êtes vous capables de deviner le type de f, g, et h, dans les trois expressions `Haskell` ci-dessus ?
2. Réécrivez, dans le langage `Haskell`, les expressions mathématiques suivantes :
   (a) $f(g(y,z),33)$,
   (b) $8g(44,23)$,
   (c) $f(8\pi(33+21))$.

Types

Exercice 5 : *Typage.* Quel est le type des expressions suivantes ?

```
['a','b','c']
('a','b','c')
[(False,'0'),(True,'1')]
([False,True],['0','1'])
[tail,init,reverse]
```

Exercice 6 : *Expressions et expressions de type.* Considérez les expressions suivantes du langage `Haskell` :

```
[Int]
[1]
[('a','b')]
[(a,b)]
['a','b']
[]
Int -> [Float]
[Int -> Float]
(Float -> Float) -> Float
Float -> Float -> Float
```

— Quelles sont les expressions de type ? Justifiez votre réponse et decrivez informellement le type décrit par l'expression.
— Si une expression n'est pas une expression de type, écrivez son type (justifiez vos réponses).

Exercice 7 : *Type et contraintes de classes des fonctions.* Considérez les définitions suivantes :

```
appl (f,x) = f x
pair x y = (x,y)
mult x y = x * y
double = mult 2
sym (x,y) = x == y
palindrome xs = reverse xs == xs
twice f x = f (f x)
incrAll xs = map (+1) xs
norme xs = sqrt (sum (map f xs)) where f x = x^2
greaters n xs = [ x | x <- xs, x > n]
menu xs = concat (map f (zip [1..length xs] xs))
    where f (n,x) = "("++show n++") " ++ show x ++ "\n"
```

1. Calculez les types de toutes ces fonctions.
2. Énumérez tous les opérateurs et/ou fonctions surchargés (c'est-à-dire, les *méthodes de classe*) qui apparaissent dans ces définitions.
3. Affinez le calcul des types en ajoutant les *contraintes de classe* au type d'une fonction, lorsque un *méthode* (ou une fonction soumise à des contraintes) apparaît dans le corps de la définition.
4. Quelles sont les fonctions de deux arguments ? Si une telle fonction n'est pas en *forme currifiée*, donnez une définition équivalente qui soit en forme currifiée.
5. Que fait la fonction `map` ? Quel est son type ?
6. Quelles sont les fonctions d'*ordre supérieur* ?
7. Quelles sont les fonctions *polymorphes* ?