Fiche de TD no. 2

Exercice 1 : *Type et contraintes de classes des fonctions.* Considérez les définitions suivantes :

```
appl (f,x) = f x
pair x y = (x,y)
sym (x,y) = x == y
palindrome xs = reverse xs == xs
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 (ignorez pour l'instant les contraintes de classe).
2. Énumérez tous les opérateurs et/ou fonctions surchargées qui apparaissent dans ces définitions (à la droite de l'égalité simple), avec leur types et contraintes de classe. Parmi ces fonctions, quelles sont des *méthodes de classe* ?
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.

Conditionnels, équations avec conditions de garde, filtrage

Exercice 2. Donnez trois possibles définitions de l'opérateur logique `(||)`, en utilisant le filtrage par motifs (rappelez vous de la définition de l'opérateur logique `&&` dans le cours).

Exercice 3. Considérez la fonction définie comme suit :

```
safetailtail xs =
    if length xs == 0 then []
    else if length xs == 1 then [] else tail (tail xs)
```

— Définissez la même fonction en utilisant, à la place des conditionnels,
  1. d'abord les équations avec conditions de garde,
  2. ensuite le filtrage par motifs,
  3. le filtrage par motifs par l'expression `case .. of ..`

— Esquissez comment les expressions conditionnelles ci-dessus se transforment en expressions `case .. of ..` lors de la précompilation du code dans le langage noyau.

Exercice 4. Dans le script de l'exercice 1 il y a au moins 3 motifs. Êtes vous capables de les reconnaître ? A quelle ligne et à quel caractère se trouvent ? Pourquoi on dit qu'il y a au moins 3 motifs et on ne dit pas qu'il y a exactement 3 motifs ?

Exercice 5. Proposez des motifs qui filtrent :

1. une liste de longueur 1 ;
2. une chaîne de caractères dont le deuxième caractère est `'a'` ;
3. une liste non vide ;
4. une liste qui contient une liste non vide ;
5. une liste avec au moins deux couples ;
6. un triplet de couples ;
7. un couple de listes, dont la deuxième n'est pas vide.

Expressions lambda

Exercice 6. Utilisez les expressions lambda pour réécrire les définitions des fonctions `norme` et `menu`—depuis l'Exercice 1—de façon à éliminer les clauses `where`.

Exercice 7. Considerez les expressions ci-dessous :

```
\n -> n + 2
\f -> \x -> f x
\f -> \xs -> [ f x | x <- xs ]
\(b,x,y) -> if b then x else y
\xs -> let m = moyenne xs in
         moyenne (map (\x -> abs (x - m)) xs)
(\n -> n + 2) 4
(\f -> \xs -> [ f x | x <- xs ]) (+1)
(\f -> \x -> f x) (\n -> n + 1)
```

Pour chaque expression :

1. expliquez ce que l'expression signifie ;
2. attribuez lui un type.

Rappel : `moyenne :: [Float] -> Float` est la fonction qui calcule la moyenne d'une liste de flottants, que vous avez défini en TP.

Compréhension sur les listes

Exercice 8. Un nombre positif est *parfait* s'il est la somme de tous ses facteurs, sauf lui même. En utilisant la compréhension, définissez[1] une fonction

```
perfects :: Int -> [Int]
```

qui retourne la liste de tous les nombres parfaits, jusqu'à la limite passée en paramètre.

Exercice 9 : *Compréhension sur les listes.* Le produit scalaire de deux listes d'entiers xs et ys de longueur n est la somme des produits des entiers correspondants :

$$xs \cdot ys = \sum_{i=0}^{n-1} xs_i * ys_i \,.$$

1. Utilisez la compréhension et les fonctions vues en cours pour définir une fonction qui retourne le produit scalaire de deux listes.
2. Autrement, utilisez l'induction pour définir cette fonction.

Exercice 10. Un triplet (x, y, z) d'entiers positifs est dit *de Pythagore* si $x^2 + y^2 = z^2$.

1. Définissez, en utilisant la compréhension, une fonction

   ```
   pyths :: Int -> [(Int,Int,Int)]
   ```

   qui envoie un entier n vers la listes de tous les triplets de Pythagore avec composantes dans `[1..n]`.
2. Si (x, y, z) est un triplet de Pythagore, alors (y, x, z) est aussi un triplet de Pythagore. Proposez une solution à l'exercice précèdent, où seulement un triplet parmi (x, y, z) et (y, x, z) apparaît dans la liste retournée.
3. Si (x, y, z) est un triplet de Pythagore, alors $x, y < z$. Proposez un solution de l'exercice précèdent qui utilise cette observation pour accélérer les calculs.

1. Vous pouvez assumer que la fonction `factors :: Int -> [Int]` (vue en cours, qui calcule les facteurs d'un nombre) est définie, donc l'utiliser pour resoudre l'exercice.