Principes de Programmation
TD1

21 janvier 2019

Exercice 1 (Paresse et Listes).
Remarque : cet exercice introduit des structures non (ou mal) vues en cours, il s'agit de les expliquer :

- la définition d'opérateurs,
- la résolution du nom comme une étape de réduction,
- les listes infinies, décrites comme `[n..]=n:[n+1..]`,
- le map,
- les fonctions via un opérateur comme `(+2)`.

1. Que fait l'opérateur `(!!)` défini dans haskell par : [1]

```
(!!) :: [a] -> Int -> a
(x:_) !! 0  =  x
(_:l) !! n  =  l !! (n-1)
[]    !! _  =  undefined
```

2. Décrivez l'évaluation de `['a','b','c','d'] !! 3`.
3. Que font les programmes suivants ? Donner leur évaluation

```
un     :: Int
un     = un

deux   :: Int
deux   = (\x -> 2) un

trois  :: Int
trois  = (\x -> x+2) un

quatre :: Int
quatre = [1, 2, 3, 4] !! 3

cinq   :: Int
cinq   = [un, 2, 3, 4, 5] !! 4
```

1. La version de la bibliothèque standard est un peu plus complèxe pour renvoyer de meilleurs message d'erreurs.

```haskell
six    :: Int
six    = [4..] !! 2

sept   :: Int
sept   = [2..]  !!  (-1)

huit   :: Int
huit   = (+2) ([4..6]  !!  2)

neuf   :: Int
neuf   = [5..9]  !!  5

dix    :: Int
dix    = map (*2) [4..]  !!  1
```

1 Chez soi

Exercice 2 (Typage et Fonctionnel (Bonus)).
Voici les λ-expressions d'opérateurs existants dans haskell :

```haskell
flip x y z      =   x z y
id              =   \x -> x
x || y          =   if x then x else y
False && _      =   False
True  && x      =   x
($)             =   \x -> \y -> x y
(.)             =   \f -> \g -> \x -> f(g x)
```

1. Que font ces opérateurs ?
2. Quels sont leurs types ?
3. Quels sont les types des fonctions suivantes :

   ```haskell
   owl      =   (.) $ (.)
   jackpot  =   ($) . ($)
   dots     =   (.) . (.)
   swing    =   flip . (. flip id)
   bigmap   =   map ($3)
   wtf      =   map $ ($) 3
   ```

4. Que font-elles ?