Ficha 2

Programação Funcional

2015/16

1. Indique como é que o interpretador de haskell avalia as expressões das alíneas que se seguem, apresentando a cadeia de redução de cada uma dessas expressões (i.e., os vários passos intermédios até se chegar ao valor final).

   (a) Considere a definição da seguinte função

   ```
   funA :: [Float] -> Float
   funA [] = 0
   funA (y:ys) = y^2 + (funA ys)
   ```

   Diga, justificando, qual é o valor de `funA [2,3,5,1]`.

   (b) Considere a definição da seguinte função

   ```
   funB :: [Int] -> [Int]
   funB [] = []
   funB (h:t) = if (mod h 2)==0 then h : (funB t)
                                else (funB t)
   ```

   Diga, justificando, qual é o valor de `funB [8,5,12]`

2. Defina recursivamente as seguintes funções sobre listas:

   (a) `dobros :: [Float] -> [Float]` que recebe uma lista e produz a lista em que cada elemento é o dobro do valor correspondente na lista de entrada.

   (b) `numOcorre :: Char -> String -> Int` que calcula o número de vezes que um caracter ocorre numa string.

   (c) `positivos :: [Int] -> Bool` que testa se uma lista só tem elementos positivos.

   (d) `soPos :: [Int] -> [Int]` que retiras todos os elementos negativos de uma lista de inteiros.

   (e) `somaNeg :: [Int] -> Int` que soma todos os números negativos da lista de entrada.

   (f) `tresUlt :: [a] -> [a]` devolve os últimos três elementos de uma lista. Se a lista de entrada tiver menos de três elementos, devolve a própria lista.

   (g) `primeiros :: [(a,b)] -> [a]` que recebe uma lista de pares e devolve a lista com as primeiras componentes desses pares.

3. Utilizando as funções `ord :: Char->Int` e `chr :: Int->Char` do módulo `Data.Char`, defina as seguintes funções:

   (a) `isLower :: Char -> Bool`
   (b) `isDigit :: Char -> Bool`
   (c) `isAlpha :: Char -> Bool`
   (d) `toUpper :: Char -> Char`
   (e) `intToDigit :: Int -> Char`
   (f) `digitToInt :: Char -> Int`

   Note que todas estas funções já estão também definidas no módulo `Data.Char` .

4. Usando as funções do módulo `Data.Char`

   (a) Defina a função `primMai`, e o seu tipo, que recebe uma string como argumento e testa se o seu primeiro caracter é uma letra maiúscula.

   (b) Defina a função `segMin`, e o seu tipo, que recebe uma string como argumento e testa se o seu segundo caracter é uma letra minúscula.

5. Recorrendo a funções do módulo `Data.Char`, defina recursivamente as seguintes funções sobre strings:

   (a) `soDigitos :: [Char] -> [Char]` que recebe uma lista de caracteres, e selecciona dessa lista os caracteres que são algarismos.

   (b) `minusculas :: [Char] -> Int` que recebe uma lista de caracteres, e conta quantos desses caracteres são letras minúsculas.

   (c) `nums :: String -> [Int]` que recebe uma string e devolve uma lista com os algarismos que occorem nessa string, pela mesma ordem.