Programação Funcional

2012/13

Mini-testes 1

1. (a) Defina a função `nome`, e o seu tipo, que recebe uma string como argumento e testa se o seu segundo caracter é uma letra minúscula.
   (Considere que a função introduzida na ficha nº1 `isLower :: Char -> Bool` já está definida).

   (b) Considere que se definiu em Haskell `Jogo` como sendo

   ```
   type Jogo = (String,Int,String,Int)
   ```

   de modo a definir resultados de jogos de futebol.
   Escreva uma função `equipaComXgolos` (e o seu tipo), que dado um jogo e o número de golos, dá como resultado o nome da equipa que marcou esses golos. Caso nenhuma equipa tenha marcado esse número de golos então a função devolve uma string vazia.

   (c) Escreva a função `jogosComXGOlos:: [Jogo] -> Int -> Int`, que dado uma lista de jogos e um número de golo, indica o número de jogos onde uma das equipa (ou as duas) marcou esse número de golos.

2. Considere os seguintes tipos para representar pontos e rectângulos, respectivamente. Assuma que os rectângulos têm os lados paralelos aos eixos e são representados apenas por dois dos pontos mais afastados.

   ```
   type Ponto = (Float,Float)
   type Rectangulo = (Ponto,Ponto)
   ```

   Defina as seguintes funções:

   - `quadrado :: Rectangulo -> Bool` que testa se um rectângulo é um quadrado.
   - `contaQuadrados :: [Rectangulo] -> Int` que, dada uma lista com rectângulos, conta quantos deles são quadrados.

3. (a) Defina a função `dist :: (Float,Float) -> Float` que dado um ponto do plano Cartesiano, calcula a sua distância à origem.

   (b) Considere a definição da seguinte função

   ```
   fun :: [Float] -> [Float]
   fun [] = []
   fun (h:t) = if h>=0 then h : (fun t)
                       else (fun t)
   ```

   Indique como é que o interpretador de haskell avalia a expressão `(fun [3,-5,0,-3,2])`, apresentando a cadeia de redução dessa expressão (i.e., os vários passos intermédios até se chegar ao valor final).

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

4. Considere os seguintes tipos para representar pontos e rectângulos, respectivamente. Assuma que os rectângulos têm os lados paralelos aos eixos e são representados apenas por dois dos pontos mais afastados.

```
type Ponto = (Float,Float)
type Rectangulo = (Ponto,Ponto)
```

Defina as seguintes funções:

- `escala :: Float -> Rectangulo -> Rectangulo` que escala um rectângulo de acordo com um dado factor (mantendo o primeiro ponto).
- `escalaTudo :: Float -> [Rectangulo] -> [Rectangulo]` que, dada um factor e uma lista com rectângulos, escala todos os rectângulos de acordo com a definição anterior.

5. (a) Defina a função `mults` que recebe três números inteiros e testa se algum deles é múltiplo dos outros dois. Não se esqueça de indicar o tipo da função.

(b) Considere a definição da seguinte função

```
fun :: [Float] -> Float
fun [] = 1
fun (x:xs) = x * (fun xs)
```

Indique como é que o interpretador de Haskell avalia a expressão `(fun [3,5,2])`, apresentando a cadeia de redução dessa expressão (i.e., os vários passos intermédios até se chegar ao valor final).

(c) Defina a função `triplos :: [Int] -> [Int]` que recebe uma lista de inteiros e produz a lista dos seus triplos.

6. (a) Considere a seguinte definição

```
p :: Int -> Bool
p 0 = True
p 1 = False
p x | x > 1 = p (x-2)
```

Qual o valor de `p 5`? Apresente as reduções que lhe permitiram chegar a essa conclusão.

(b) Considere as seguintes definições de tipo para representar círculos (guardando o centro e raio)

```
type Ponto = (Float, Float) -- (Abcissa, Ordenada)
type Circulo = (Ponto, Float) -- (Centro, Raio)
```

i. Defina uma função `dentro :: Ponto -> Circulo -> Bool` que testa se um ponto está dentro de um círculo (i.e., se a distância ao centro é menod do que o raio).

ii. Defina uma função `filtra :: Ponto -> [Circulo] -> Int` que, dado um ponto e uma lista de círculos, determina quantos círculos da lista contêm o ponto dado (use, se precisar, a função anterior).

7. (a) Defina a função `somaIgual`, e o seu tipo, que recebe três números inteiros e testa se a soma de dois desses argumentos é igual ao valor do argumento restante.

(b) Considere que se definiu em Haskell `Jogo` como sendo

```
type Jogo = (String,Int,String,Int)
```

de modo a definir resultados de jogos de futebol.

Escreva uma função `resJogo` (e o seu tipo), que dado um jogo devolve um character com valor `'1'` (indicando que ganhou a equipa da casa), `'x'` (indicando empate) e `'2'` (vitário da equipa visitante). Por exemplo, `resJogo ("Alemanha",2,"Portugal",3)` dá como resultado `'2'`.

(c) Escreva uma função, e o seu tipo, que dado uma lista de jogos indica quantos jogos venceram as equipadas visitantes.

8. (a) Defina a função `supSoma` que recebe três números inteiros e testa se algum deles é superior à soma dos outros dois. Não se esqueça de indicar o tipo da função.

(b) Considere a definição da seguinte função

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

Indique como é que o interpretador de Haskell avalia a expressão `(fun [2,3,5])`, apresentando a cadeia de redução dessa expressão (i.e., os vários passos intermédios até se chegar ao valor final).

(c) Defina a função `soDigitos :: [Char] -> [Char]` que recebe uma lista de caracteres, e selecciona dessa lista os caracteres que são algarismos. Relembre que a função `isDigit::Char -> Bool`, já pré-definida, testa se um caracter representa um algarismo.

9. Considere os seguintes tipos para representar pontos e rectângulos, respectivamente. Assuma que os rectângulos têm os lados paralelos aos eixos e são representados apenas por dois dos pontos mais afastados.

```
type Ponto = (Float,Float)
type Rectangulo = (Ponto,Ponto)
```

Defina as seguintes funções:

- `area :: Rectangulo -> Float` que determina a área de um rectângulo.
- `areaTotal :: [Rectangulo] -> Float` que, dada uma lista com rectângulos, determina a área total que eles ocupam.

10. (a) Considere a seguinte definição

```
p :: Int -> Bool
p 0 = True
p 1 = False
p x | x > 1 = p (x-2)
```

Qual o valor de `p 5`? Apresente as reduções que lhe permitiram chegar a essa conclusão.

(b) Considere as seguintes definições de tipo para representar círculos (guardando o centro e raio)

```
type Ponto = (Float, Float) -- (Abcissa, Ordenada)
type Circulo = (Ponto, Float) -- (Centro, Raio)
```

i. Defina uma função `fora :: Ponto -> Circulo -> Bool` que testa se um ponto está fora de um círculo (i.e., se distância ao centro é maior do que o raio).

ii. Defina uma função `filtra :: Circulo -> [Ponto] -> Int` que, dado um círculo e uma lista de pontos, determina quantos pontos da lista estão fora do círculo dado (use, se precisar, a função anterior).

11. (a) Defina a função `mult` de forma a que (`mult` n a b) teste se n é simultaneamente múltiplo de a e de b. Não se esqueça de indicar o tipo da função.

    (b) Considere a definição da seguinte função

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

    Indique como é que o interpretador de Haskell avalia a expressão `(fun [8,5,12,7])`, apresentando a cadeia de redução dessa expressão (i.e., os vários passos intermédios até se chegar ao valor final).

    (c) Defina a função `minusculas :: [Char] -> Int` que recebe uma lista de caracteres, e conta quantos desses caracteres são letras minúsculas. Relembre que a função `isLower::Char -> Bool`, já pré-definida, que testa se um caracter representa uma letra minúscula.

12. (a) Defina a função `nome`, e o seu tipo, que recebe uma string como argumento e testa se o seu primeiro caracter é uma letra maiúscula.
    (Considere que a função introduzida na ficha nº1 `isUpper :: Char -> Bool` já está definida).

    (b) Considere que se definiu em Haskell `Jogo` como sendo

    ```
    type Jogo = (String,Int,String,Int)
    ```

    de modo a definir resultados de jogos de futebol.
    Escreva uma função `golosEquipa` (e o seu tipo), que dado um jogo e o nome de uma equipa, dá como resultado os golos que essa equipa marcou. Caso a equipa não tenha participado no jogo então a função devolve o valor `-1`.

    (c) Escreva a função `golos:: [Jogo] -> String -> Int`, que dado uma lista de jogos e o nome de uma equipa, indica quantos golos essa equipa marcou em todos os jogos.

13. Considere os seguintes tipos para representar pontos e rectângulos, respectivamente. Assuma que os rectângulos têm os lados paralelos aos eixos e são representados apenas por dois dos pontos mais afastados.

    ```
    type Ponto = (Float,Float)
    type Rectangulo = (Ponto,Ponto)
    ```

    Defina as seguintes funções:

    - `roda :: Rectangulo -> Rectangulo` que roda um rectângulo 90º (centrado no primeiro ponto).
    - `rodaTudo :: [Rectangulo] -> [Rectangulo]` que, dada uma lista com rectângulos, roda todos os rectângulos de acordo com a definição anterior.

14. (a) Defina a função `maior`, e o seu tipo, que recebe três números inteiros e testa se algum deles é maior que a soma dos outros dois.

    (b) Considere que se definiu em Haskell `Jogo` como sendo

```
type Jogo = (String,Int,String,Int)
```

de modo a definir resultados de jogos de futebol. Escreva uma função `resJogo` (e o seu tipo), que dado um jogo devolve uma string que indica se ganhou a equipa da casa, a equipa visitante, ou se empataram.

Por exemplo, `resJogo ("Alemanha",2,"Portugal",3)` dá como resultado a string `"ganhou equipa visitante"`.

(c) Escreva uma função, e o seu tipo, que dado uma lista de jogos indica quantos jogos terminaram em empate.