Programação Funcional

2012/13

Mini-testes 3

1. Considere que a GNR, para registar os teste de alcoolémia que realiza nas suas operações stop, utiliza a seguinte estrutura de dados:

```
type RegAlcool = [(Nome,Sexo,Idade,NA)]
type Nome  = String
type Sexo  = Char      -- 'M': Masculino 'F': Feminino
type Idade = Int
type NA    = Float     -- Nivel de Alcool
```

   (a) Escreva uma função de ordem superior que dados os testes realizados, devolve informação dos testes realizados a pessoas menores de 21 anos.

   (b) Considerando que a multa a pagar para quem conduzir com uma taxa no sangue superior a `0.5` se calcula de acordo com a regra `Nível de Alcoól * 100 EUR`, escreva uma função que dados os testes realizados, produz uma lista onde se indica: o nome, a idade e o valor da multa a pagar (`0 EUR` caso esteja legal).

   (c) De modo a fazer uma estimativa da idade média das pessoas que realizaram o teste, escreva uma função de ordem superior que cálcula esse valor.

2. (a) Apresente uma definição alternativa da seguinte função, usando recursividade explícita em vez de funções de ordem superior.

```
func :: Eq a => a -> [a] -> Bool
func x l = not (null (filter (x==) l))
```

   (b) Como vimos nas aulas, uma matriz pode ser representada pelo seguinte tipo:

```
type Matriz a = [[a]]
```

   Defina a função `zero :: Matriz a -> Bool`, que testa se uma matriz contém apenas zeros, sem usar recursividade explícita.

3. Uma forma de representar polinómios de uma variável é usar listas de monómios representados por pares *(coeficiente, expoente)*

```
type Polinomio = [Monomio]
type Monomio = (Float,Int)
```

   Por exemplo, `[(2,3), (3,4), (5,3), (4,5)]` representa o polinómio $2\,x^3 + 3\,x^4 + 5\,x^3 + 4\,x^5$.

   (a) Defina uma função `conta :: Int -> Polinomio -> Int` de forma a que `(conta n p)` indica quantos monómios de grau `n` existem em `p`.

(b) Defina uma função `selgrau :: Int -> Polinomio -> Polinomio` de forma a que `(selgrau n p)` que selecione do polinómio `p` os monómios de grau superior a `n`. De preferência, use funções de ordem superior.

(c) Complete a definição da função `deriv` de forma a que esta calcule a derivada de um polinómio.

```
deriv :: Polinomio -> Polinomio
deriv p = map ........ p
```

4. Considere que a GNR desenvolveu um radar portátil para instalar nas suas viaturas de modo a detectar excessos de velocidade na estrada. Este radar usa a seguinte estrutura de dados para registar excessos de velocidade num dia:

```
type Radar = [(Hora,Matricula,VelAutor,VelCond)
type Hora = (Int,Int)
type Matricula = String    -- matricula do carro em infraccao
type VelAutor = Int        -- velocidade autorizada
type VelCond  = Float      -- velocidade do condutor
```

(a) Utilizando a função de ordem superior `filter`, escreva uma função que verifica se o radar está a funcionar correctamente (isto é, a velocidade do condutor é sempre maior que a velocidade autorizada).

(b) Escreva uma função que dado os registos de infrações de um dia, devolve a lista com a matricula do carro e o excesso de velocidade a que se deslocava. O excesso de velocidade para um registo é a diferença entre velocidade real (após aplicar a tolerência) e velocidade autorizada.

(c) Escreva uma função de ordem superior que dado a lista produzida na alínea anterior, calcula o total de escesso de velocida.

5. (a) Apresente uma definição alternativa da seguinte função, usando recursividade explícita em vez de funções de ordem superior.

```
func :: [[a]] -> [Int]
func l = map length (filter null l)
```

(b) Como vimos no mini-teste passado, um multi-conjunto pode ser representado pelo seguinte tipo.

```
type MSet a = [(a,Int)]
```

Defina a função `elem :: Eq a => a -> MSet a -> Bool`, que testa se um elemento pertence a um multi-conjunto, sem usar recursividade explícita.

6. Considere a seguinte definição usando listas por compreensão:

```
prod :: [a] -> [b] -> [(a,b)]
prod l1 l2 = [(a,b) | a <- l1, b <- l2]
```

Por exemplo,

```
prod [1,2] ['a','b','c'] = [(1,'a'),(1,'b'),(1,'c'),(2,'a'),(2,'b'),(2,'c')]
```

De forma a definir esta função sem usar o mecanismo de definição de listas por compreensão, vamos defini-la à custa de outras funções auxiliares:

(a) A função `criaPares :: a -> [b] -> [(a,b)]` recebe um elemento do tipo `a` e uma lista e cria uma lista de pares cuja primeira componente é sempre a mesma.
Por exemplo, `criaPares 1 ['a','b','c'] = [(1,'a'),(1,'b'),(1,'c')]`

- Apresente uma definição (explicitamente) recursiva da função `criaPares`.
- Complete a seguinte definição da função `criaPares` (comece por determinar o tipo da função `acrescenta`)

```
criaPares a bs = map (acrescenta a) bs
    where acrescenta a x = ...
```

(b) Finalmente, podemos definir a função pretendida, concatenando todas as linhas produzidas pela função anterior.

```
prod l1 l2 = concat (criaLinhas l1 l2)
```

Apresente uma definição da função `concat :: [[a]] -> [a]` que concatena uma lista de listas numa lista só.

7. Uma forma de representar polinómios de uma variável é usar listas de monómios representados por pares *(coeficiente, expoente)*

```
type Polinomio = [Monomio]
type Monomio = (Float,Int)
```

Por exemplo, `[(2,3), (3,4), (5,3), (4,5)]` representa o polinómio $2\,x^3 + 3\,x^4 + 5\,x^3 + 4\,x^5$.

(a) Defina uma função `calcula :: Float -> Polinomio -> Float`, que calcula o valor do polinómio num dado valor de x.

(b) Defina a função `simp :: Polinomio -> Polinomio` que retira de um polinómio os monómios de coeficiente zero. De preferência, use funções de ordem superior.

(c) Complete a definição da função `mult` de forma a que esta calcule o resultado da multiplicação de um monómio por um polinómio.

```
mult :: Monomio -> Polinomio -> Polinomio
mult (c,e) p = map ........ p
```

8. Considere que a GNR, para registar os teste de alcoolémia que realiza nas suas operações stop, utiliza a seguinte estrutura de dados:

```
type RegAlcool = [(Nome,Sexo,Idade,NA)]
type Nome  = String
type Sexo  = Char      -- 'M': Masculino 'F': Feminino
type Idade = Int
type NA    = Float     -- Nivel de Alcool
```

(a) Escreva uma função de ordem superior que dados os testes realizados, devolve os testes realizados a mulheres apenas.

(b) Considerando que a lei autoriza a condução com uma taxa de alcoól no sangue de `0.5`, escreva uma função que dados os testes realizados, produz uma lista onde se indica: o nome da pessoa (que fez o teste) e uma string a indicar se a condução é ``legal'' ou ``ilegal''.

(c) De modo a fazer uma estimativa do montante correspondente às várias multas passados, pretende-se escrever uma função de ordem superior que dados os testes realizados calcula o seu valor de acordo com a regra: `Nível de Alcoól * 100 EUR` no caso do nível de alcoól for superior ao permitido.

9. (a) Apresente uma definição alternativa da seguinte função, usando recursividade explícita em vez de funções de ordem superior.

```
func :: [a] -> [a] -> Bool
func l m = and (zipWith (==) l m)
```

(b) Como vimos nas aulas, uma matriz pode ser representada pelo seguinte tipo:

```
type Matriz a = [[a]]
```

Defina a função `quadrada :: Matriz a -> Bool`, que testa se uma matriz é quadrada, sem usar recursividade explícita.

10. Considere que a GNR desenvolveu um radar portátil para instalar nas suas viaturas de modo a detectar excessos de velocidade na estrada. Este radar usa a seguinte estrutura de dados para registar excessos de velocidade num dia:

```
type Radar = [(Hora,Matricula,VelAutor,VelCond)
type Hora = (Int,Int)
type Matricula = String    -- matricula do carro em infraccao
type VelAutor = Int        -- velocidade autorizada
type VelCond  = Float      -- velocidade do condutor
```

(a) Geralmente a GNR considera que a velocidade registada do condutor pode ter um erro de 10% (em excesso). Escreva uma função de ordem superior que dado as infrações registadas pelo radar num dia, aplica esta tolerância á velocidade registada.

(b) Escreva uma função de ordem superior que dado a matricula de um carro e as insfrações registadas num dia, devolve uma lista com as infrações desse carro. Note que um carro pode ser apanhado em excesso mais do que uma vez no mesmo dia.

(c) Escreva um função de ordem superior que calcula o total do excesso de velocidade num dia. O excesso de velocidade para um registo é a diferença entre velocidade real (após aplicar a tolerência) e velocidade autorizada.

11. Assuma que a informação sobre os resultados dos jogos de uma jornada de um campeonato de futebol está guardada na seguinte estrutura de dados:

```
type Jornada = [Jogo]
type Jogo = ((Equipa,Golos),(Equipa,Golos))     -- (eq.casa, eq. visitante)
type Equipa = String
type Golos = Int
```

(a) Defina a função `totalGolos :: Jornada -> Int` que calcula o total de golos da jornada.

(b) Defina a função `numGolos :: Int -> Jornada -> [Jogo]` de forma a que `(numGolos x j)` represente a lista de jogos com mais de `x` golos marcados. De preferência, use funções de ordem superior.

(c) Considere a seguinte função:

```
venceCasa :: Jornada -> [Equipa]
venceCasa j = map casa (filter vc j)
```

Defina as funções `vc` e `casa` de forma a que a função `venceCasa` calcule a lista das equipas que venceram em casa numa dada jornada.

12. Considere a seguinte definição usando listas por compreensão:

```
prod :: [a] -> [b] -> [(a,b)]
prod l1 l2 = [(a,b) | a <- l1, b <- l2]
```

Por exemplo,

```
prod [1,2] ['a','b','c'] = [(1,'a'),(1,'b'),(1,'c'),(2,'a'),(2,'b'),(2,'c')]
```

De forma a definir esta função sem usar o mecanismo de definição de listas por compreensão, vamos defini-la à custa de outras funções auxiliares:

(a) A função `criaLinhas :: [a] -> [b] -> [[(a,b)]]` aplica a função anterior a cada elemento da primeira lista.
Por exemplo,

```
criaLinhas [1,2] ['a','b','c'] =
[[(1,'a'),(1,'b'),(1,'c')],[(2,'a'),(2,'b'),(2,'c')]]
```

- Apresente uma definição (explicitamente) recursiva da função `criaLinhas`.
- Complete a seguinte definição da função `criaLinhas` (comece por determinar o tipo da função `f`)

```
crialinhas l1 l2 = map (f l2) l1
    where f l x = ...
```

(b) Finalmente, podemos definir a função pretendida, concatenando todas as linhas produzidas pela função anterior.

```
prod l1 l2 = concat (criaLinhas l1 l2)
```

Apresente uma definição da função `concat :: [[a]] -> [a]` que concatena uma lista de listas numa lista só.

13. (a) Apresente uma definição alternativa da seguinte função, usando recursividade explícita em vez de funções de ordem superior.

```
func :: Ord a => a -> [a] -> Int
func x l = length (filter (>= x) l)
```

(b) Como vimos no mini-teste passado, um multi-conjunto pode ser representado pelo seguinte tipo.

```
type MSet a = [(a,Int)]
```

Defina a função `size :: MSet a -> Int`, que determina o tamanho de um multi-conjunto, sem usar recursividade explícita.

14. Assuma que a informação sobre os resultados dos jogos de uma jornada de um campeonato de futebol está guardada na seguinte estrutura de dados:

```
type Jornada = [Jogo]
type Jogo = ((Equipa,Golos),(Equipa,Golos))
type Equipa = String
type Golos = Int
```

(a) Defina a função `pontos :: Jornada -> [(Equipa,Int)]` que calcula os pontos que cada equipa obteve na jornada (venceu - 3 pontos; perdeu - 0 pontos; empatou - 1 ponto)

(b) Defina a função `empates :: Jornada -> [Jogo]` que seleciona os jogos da jornada em que ocorreram empates. De preferência, use funções de ordem superior.

(c) Considere a seguinte função:

```
golosMarcados :: Jornada -> Int
golosMarcados j = sum (map soma j)
```

Defina a função `soma` de forma a que a função `golosMarcados` calcule o número total de golos marcados numa jornada.