Aula Teórico-prática e Prática 7

Programação Funcional

LCC 1º ano

Considere os seguintes tipos para representar polinómios de coeficientes inteiros (de uma variável)

```
type Poly1 = [(Coeficiente,Grau)]
type Poly2 = [Coeficiente]
type Coeficiente = Int
type Grau = Int
```

No tipo `Poly2` é necessário garantir que guardamos os coeficientes por ordem crescente do grau e que incluímos os coeficientes nulos. Por uma questão de simplicidade, vamos também assumir que a primeira representação também guarda os monómios por ordem crescente do grau. Assim, o polinómio

$$3.x^5 + 2.x^2 + 4.x + 5$$

é representado em Haskell por

```
p1 :: Poly1
p1 = [(5,0), (4,1), (2,2), (3,5)]
p2 :: Poly2
p2 = [5,4,2,0,0,3]
```

1. Defina funções de conversão entre estas duas representações.
2. Defina a função `addP2 :: Poly2 -> Poly2 -> Poly2` que adiciona dois polinómios. Use essa função bem como as funções de conversão para obter uma função de adição de polinómios na representação `Poly1`.
3. Defina uma função `valor :: Poly2 -> Int -> Int` que calcula o valor de um polinómio num ponto.
4. Use a função anterior para definir a função `atoi :: String -> Int` que converte uma string com os dígitos de um número no número inteiro correspondente.

   Note que a string "300245" representa o número

   $$3.10^5 + 2.10^2 + 4.10 + 5$$

   que não é nada mais do que o valor do polinómio

   $$3.x^5 + 2.x^2 + 4.x + 5$$

   no ponto 10.
5. Defina a função inversa da anterior (considere apenas números positivos).