Aula Teórico-Prática 2

Programação Funcional

LEI 1º ano

Funções não recursivas

Exercícios

1. Utilisando as funções `ord::Char -> Int` e `chr::Int -> Char` defina as seguintes funções:

   ```
   isLower ::  Char -> Bool
   isDigit ::  Char -> Bool
   isAlpha ::  Char -> Bool
   toUpper ::  Char -> Char
   ```

2. Defina uma função `max2` que calcula o maior de dois números inteiros. Comece por definir a assinatura da função.

3. Defina uma função que calcula o maior de três números inteiros. Para isso apresente duas definições alternativas: recorrendo ou não à função `max3` definida na alínea anterior.

4. Num triângulo verifica-se sempre que a soma dos comprimentos de dois dos lados é superior (ou igual) à do terceiro. A esta propriedade chama-se *desigualdade triangular*. Defina uma função que, dados três números, teste se esses números correspondem aos comprimentos dos lados de um triângulo.

5. Considere a seguinte definição:

   ```
   opp :: (Int,(Int,Int)) -> Int
   opp z = if ((fst z) == 1)
           then (fst (snd z)) + (snd (snd z))
           else if ((fst z) == 2)
                then (fst (snd z)) - (snd (snd z))
                else 0
   ```

   Apresente uma definição alternativa que use concordância de padrões em vez dos *if*s.

6. Defina uma função que recebe os (3) coeficientes de um polinómio de 2º grau e que calcula o número de raízes (reais) desse polinómio.

7. Usando a função anterior, defina uma função que, dados os coeficientes de um polinómio de 2º grau, calcula a lista das suas raízes reais.

8. As funções das duas alíneas anteriores podem receber um tuplo com os coeficientes do polinómio, ou receber os 3 coeficientes separadamente. Defina a versão alternativa ao que definiu acima.