Disciplina de Programação Funcional
2ª Lista de Exercícios
Curso de Engenharia de Computação
UEMG Ituiutaba
https://bit.ly/2A0eUl8
https://github.com/mauro-hemerly/UEMG-2018-2

1. Teste as seguintes expressões no sistema WinGHCi, e informe o tipo de cada uma:
   - `[]`
   - `[10, -7, 11, 19]`
   - `[0.6, 0.75, 0.9]`
   - `['a', 'b', 'c', 'd']`
   - `[True, False, False]`
   - `["maio", "abril"]`
   - `[(12,'a'),(13,'b')]`
2. Teste as seguintes expressões no sistema WinGHCi:
   (a) `5:[] == [5]`
   (b) `'a':['n','a'] == ['a','n','a'] == "ana"`
   (c) `1:[2,3] ++ [4,5,6] == [1,2,3,4,5,6]`
   (d) `1:2:3:[] == [1,2,3]`
   (e) `head [1,3,5,7,9,20]`
   (f) `tail [1,3,5,7,9,20]`
   (g) `last [1,3,5,7,9,20]`
   (h) `init [1,3,5,7,9,20]`
3. Teste as seguintes expressões no sistema WinGHCi, e informe o tipo de cada uma:
   (a) `[1,2,3]`
   (b) `[0,2,'a']`
   (c) `[1,2,3] == [2,1,3]`
   (d) `[1 .. 60]`
   (e) `[1,4 .. 60]`
   (f) `[0.1,0.2 .. 0.9]`
   (g) `[0.1 .. 9]`
   (h) `["apartamento",['c','a','s','a']]`
   (i) `[('a',3),('b',4),('c',5)]`
   (j) `['a','b',['c','d']] /= ['a','b','c','d']`
   (k) `sum [1..10]`
4. Teste a função **conta** abaixo:

   ```
   conta :: [Int] -> Int
   conta [] = 0
   conta (x:xs) = 1 + conta xs
   ```

   Em seguida faça os seguintes testes:

   ```
   conta [1,2,3]
   conta [1..50]
   conta [0.01, 0.02 .. 0.10]
   conta [(0,'a'),(1,'b'),(2,'c')]
   ```

   Quais os resultados? Como modificar a função para que possamos obter o comprimento de uma lista com elementos de qualquer tipo? Por exemplo, lista de números reais, palavras, tuplas? Faça esta modificação e teste novamente a função.
5. Escreva uma função recursiva que verifica se um dado elemento pertence à uma lista. A função deve ter como entrada: uma lista e o elemento a ser procurado, e deve retornar como resultado um valor booleano (True/False) como no exemplo abaixo.

   ```
   Prelude> pertence 'a' ['b','c','a','d']
   True
   ```

6. Escreva uma função para calcular a média dos elementos de uma lista de números. Podem-se usar duas funções, uma para obter a quantidade de elementos e outra para obter a soma dos elementos, e finalmente, calcular a divisão entre a soma e a quantidade.

   ```
   Prelude> media [4,5,2,7]
   4.5
   ```

7. Teste a função abaixo para retornar o maior elemento de uma lista:

   ```
   maior [a] = a
   maior (a : t) = if (a > (maior t)) then a
     else (maior t)
   ```

(a) Reescreva a função eliminando o comando if.

(b) Baseado na função **maior**, faça uma nova função que devolva o menor valor de uma lista.

(c) Usando as funções **maior** e **menor**, crie uma nova função que dada uma lista, retorne o maior e menor elemento numa **tupla-2**.

8. No módulo **Char** encontramos a função **toUpper** que converte uma letra **minúscula** na sua correspondente **maiúscula**. Para que possamos converter todas as letras de uma palavra em maiúsculas podemos escrever:

```
import Data.Char
maiuscula::[Char]->[Char]
maiuscula [] = []
maiuscula (a:as) = toUpper a: maiuscula as
```

(a) Teste a função para a palavra "aBC1cde".

(b) Usando a função **isAlpha** exemplificada abaixo, refaça a função **maiuscula** para descartar símbolos ou números.

```
Prelude> isAlpha 'b'
True
Prelude> isAlpha '1'
False
Prelude> isAlpha '&'
False
```

(c) Faça uma nova função que recebe uma palavra e retorne numa tupla-2 a palavra original e a sua correspondente escrita em maiúsculas.

9. Escrever uma função **listUp** que cria uma lista de inteiros de 1 até o valor passado como parâmetro.