Programação Funcional
Folha de Exercícios 01
Funções do Prelude e Compreensão de Listas

Prof. Wladimir Araújo Tavares

1. Defina cada uma das seguintes funções usando apenas funções pré-definidas do módulo Prelude:

   (a) Defina a função interior tal que (`interior xs`) é uma lista obtida eliminando os extremos da lista xs. Por exemplo,
   ```
   interior [2,5,3,7,3] == [5,3,7]
   ```

   (b) Defina a função final tal que (`final xs`) é uma lista formada pelos n elementos finais de xs. Por exemplo,
   ```
   final 3 [2,5,4,7,9,6] == [7,9,6]
   ```

   (c) Defina uma função segmento tal que (segmento n m xs) é uma lista dos elementos de xs compreendidos entre as posições n e m. Por exemplo,
   ```
   segmento 2 3 [3,4,1,2,7,9,0] == [1,2]
   ```

   (d) Defina uma função extremos tal que (extremos n xs) é uma lista formada pelos n primeiros elementos de xs e n finais elementos de xs. Por exemplo,
   ```
   extremos 3 [2,6,7,1,2,4,5,8,9,2,3] == [2,6,7,9,2,3]
   ```

   (e) Escreva uma função `dotprod :: [Float] -> [Float] -> Float` para calcular o produto interno de dois vetores (representados como listas): $dotprod[x_1,\ldots,x_n][y_1,\ldots,y_n] = x_1 * y_1 + \ldots + x_n * y_n = \sum_{i=1}^{n} x_i * y_i$
   Usando a função `zipWith` e `sum`.

   (f) Escreva a definição da função `elemento :: Eq a => a -> [a] -> Bool` que testa se uma valor ocorre em uma lista usando a função `any :: (a -> Bool) -> [a] -> Bool`. Por exemplo,
   ```
   any (==1) [1,2,3] == True
   ```

2. Defina as seguintes funções usando compreensão de listas:

   (a) Defina a função `somaQuadrados :: Integer -> Integer` tal que somaQuadrados n é a soma dos quadrados dos n primeiros números, ou seja, $1^2 + 2^2 + 3^2 + \ldots + n^2$. Por exemplo,
   ```
   somaQuadrados 3 == 14
   ```

   (b) Defina a função `replica :: Int -> a -> [a]` tal que replica n xs é uma lista formada por n cópias do elemento x. Por exemplo,
   ```
   replica 3 True == [True, True, True]
   ```

   (c) Os triângulos aritméticos se formam da seguinte maneira:

   ```
   1
   2    3
   4    5    6
   7    8    9    10
   11   12   13   14   15
   16   17   18   19   20   21
   ```

   Defina uma função linha tal que linha n é a n-ésima linha do triângulo aritmético. Por exemplo,
   ```
   linha 4 == [7,8,9,10]
   ```

   (d) Defina a função triângulo tal que triangulo n é um triângulo aritmético de altura n. Por exemplo,
   ```
   triangulo 3 == [ [1], [2,3], [4,5,6] ]
   ```

   (e) Defina uma função fatores tal que fatores n é uma lista formada pelos divisores de n. Por exemplo,
   ```
   fatores 6 == [1,2,3,6]
   ```

   (f) Um inteiro positivo é perfeito se ele é igual a soma dos seus fatores, excluindo ele mesmo. Por exemplo, 6 é igual 1 + 2 + 3. Defina a função `perfeitos :: Int -> [Int]` tal que (perfeitos n) é uma lista de todos os números perfeitos menores que n. Por exemplo,
   ```
   perfeitos 500 == [6,28,496]
   ```

   (g) Defina uma fatoresPrimos tal que fatoresPrimos n devolve uma lista de tuplas (f, b) representando os fatores primos de n onde f é um fator primo de n e b é o seu expoente. Todo número x, tal que $x \in \mathbb{N}$, pode ser escrito como o produto de potências de bases primas e expoentes naturais. Por exemplo, o número 33661743 pode ser reescrito na forma,

   $$3361743 = 3^4 \times 7^3 \times 11^2 \tag{1}$$

   Os números 3, 7 e 11 são denominados fatores primos de 3361743 e 4, 3 e 2 seus respectivas expoentes.)
   PROT:
   ```
   factors ::  (Integral a) => a -> [(a,a)]
   ```
   EX(S):
   ```
   factors 3361743 => [(3,4),(7,3),(11,2)]
   ```

   (h) `interseccao :: Eq a => [a] -> [a] -> [a]` tal que (interseccao xs ys) é a interseccao dos conjuntos xs e ys. Por exemplo,
   ```
   interseccao [3,2,5] [5,7,3,4] == [3,5]
   ```

   (i) `diferenca :: Eq a => [a] -> [a] -> [a]` tal que (diferenca xs ys) é a diferença entre os conjuntos xs e ys. Por exemplo,
   ```
   diferenca [3,2,5,6] [5,7,3,4] == [2,6]
   diferenca [3,2,5] [5,7,3,2] == []
   ```

   (j) Escreva uma função diamonds que receba como argumento um valor inteiro positivo n e retorne uma lista de listas $[l_1, l_2, \ldots, l_{n-1}, l_n, l_{n-1}, \ldots, l_2, l_1]$, onde l_i é a lista dos i múltiplos de i (sendo i o primeiro múltiplo de i).
   Exemplo:
   ```
   diamonds 2 [[1],[2,4],[1]]
   diamonds 3 [[1],[2,4],[3,6,9],[2,4],[1]]
   diamonds 4 [[1],[2,4],[3,6,9],[4,8,12,16],[3,6,9],[2,4],[1]]
   ```

   (k) A função `aplica :: [a -> a] -> a -> [a]` recebe uma lista de funções e um valor retornando uma lista com os resultados das aplicações das funções. Por exemplo,
   ```
   aplica [(*4), (+2), (-4)] 2 == [8,4,-2]
   ```

   (l) Um trio (x, y, z) de inteiros positivos diz-se pitagórico se $x^2 + y^2 = z^2$. Defina a função `pitagoricos :: Int -> [(Int, Int, Int)]` que calcule todos os trios pitagóricos cujas componentes não ultrapassem o argumento.
   Por exemplo: `pitagoricos 10 = [(3, 4, 5), (4, 3, 5), (6, 8, 10), (8, 6, 10)]`.

   (m) Os polinômios podem ser representados da forma dispersa ou densa. Por exemplo, o polinômio $6x^4 - 5x^2 + 4x - 7$ pode ser representado da forma dispersa por [6,0,-5,4,-7] e da forma densa [(6,4), (-5, 2), (4, 1), (-7, 0)]. Defina uma função `densa :: [Int] -> [(Int,Int)]` tal que densa xs é a representação densa do polinômio cuja representação dispersa é xs. Por exemplo,
   ```
   densa [6,0,-5,4,-7] == [(4,6),(2,-5),(1,4),(0,-7)]
   densa [6,0,0,3,0,4] == [(5,6),(2,3),(0,4)]
   ```

   (n) Defina a função neglist xs que computa o número de elementos negativos em uma lista xs.
   ```
   neglist [1,2,3,4,5] == 0
   neglist [1,-3,-4,3,4,-5] == 3
   ```

   (o) Defina uma função `gensquares low high` que gera uma lista de todos os quadrados de todos os números pares a partir de um limite inferior low até um limite superior high.
   ```
   gensquares 2 5 == [4,16]
   gensquares 3 10 == [16,36,64,100]
   ```

3. Considere a seguinte série (i.e. somas infinitas) que convergem para π:

   $$\pi = \frac{4}{1} - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \ldots \tag{2}$$

   (a) Construa uma lista infinita com os numerados das parcelas.
   $[4, -4, \ldots, 4(-1)^n]$

   (b) Construa uma lista infinita com os denominadores das parcelas.

   (c) Combine as duas listas usando a função `zipWith`.

   (d) Defina a função `calcPi1` n que calcula o valor de pi somando n parcelas da série. Calcule o valor o somatório para 10, 100 e 1000 parcelas.

   (a) Escreva uma função binom com dois argumentos que calcule o coeficiente

   $$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3}$$

   Sugestão: pode exprimir n! como product [1..n]

   (b) Usando uma função binom que calcula o coeficiente binomial, escreva uma definição da função `pascal :: Int -> [[Int]]` que calcula as primeiras linhas triângulo de Pascal.

   $$\binom{0}{0}$$

   $$\binom{1}{0} \quad \binom{1}{1}$$

   $$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

   $$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

   ```
   pascal 4 == [[1],[1,1],[1,2,1],[1,3,3,1],[1,4,6,4,1]]
   ```

   (c) defina o triângulo de Pascal completo como uma lista infinita `pascal2 :: [[Int]]` das linhas do triângulo. Note que que `pascal2 !! n !! k = binom n k`, para quaisquer n e k tais que $n \geq 0$ e $0 \leq k \leq n$.
   Exemplos:
   ```
   pascal2 !! 6 == [1,6,15,20,15,6,1]
   pascal2 !! 6 !! 3 == 20
   ```