Os exercícios devem ser entregues em um arquivo obrigatoriamente com o nome **Lista1.hs**. Os nomes das funções devem ser idênticos aos nomes apresentados nos exemplos.

Serão considerados apenas arquivos que estejam sendo carregados corretamente (sem erros) pelo interpretador.

1. Declare uma função que receba as 3 medidas dos lados de um triangulo, a função deve informar se as medidas podem formar um triângulo Retornando a True em caso afirmativo e False caso contrário, por exemplo:

   ehTriangulo 10 20 30 => False
   ehTriangulo 5 3 3 => True,
   ehTriangulo 5 3 4 => True

2. Declare uma função que receba 3 medidas válidas dos lados de um triângulo e retorne se esse triângulo é equilátero, isósceles ou escaleno. O retorno deve ser uma String contendo a classificação do triângulo, por exemplo:

   tipoTriangulo 5 5 5 => "equilatero"
   tipoTriangulo 5 3 3 => "isosceles"
   tipoTriangulo 5 3 4 => "escaleno"

3. **Usando as funções declaradas nos exercícios anteriores** defina uma função que receba as 3 medidas dos lados de um triângulo e retorne se essas medidas formam um triângulo, em caso afirmativo a função deve retornar o tipo do triângulo: equilátero, isósceles ou escaleno, caso contrário deve retornar a string: "não eh um triangulo".

   triangulo 5 5 5 => "equilatero"
   triangulo 5 3 3 => "isosceles"
   triangulo 5 3 4 => "escaleno"
   triangulo 15 6 5 => "nao eh um triangulo"

4. Declare uma função que receba como parâmetro um inteiro *n* e retorne a soma dos números pares entre 0 e *n*.

   somaPares 5 => 4 + 2 + 0 = 6
   somaPares 8 => 8 + 6 + 4 + 2 + 0 = 20

5. Declare uma função que receba inteiros (*m* e *n*) e retorne a seguinte série: $2^0m + 2^1m + 2^2m + \ldots + 2^nm$. Por exemplo:

   somaPot2m 6 4 => 6 + 12 + 24 + 48 + 96 = 186
   somaPot2m 3 3 => 3 + 6 + 12 + 24 = 45

6. Declare uma função que receba um número e retorne True caso o número seja primo e False caso contrário. Um número primo é um número natural maior que 1, e que possui apenas dois divisores: 1 e ele mesmo. Por exemplo

   primo 37 => True
   primo 10 => False

7. Uma aproximação para o valor de π pode ser obtida por meio da série:

   4/1 – 4/3 + 4/5 – 4/7 + 4/9 – 4/11 + ...

   Declare uma função chamada seriePI que receba como parâmetro um inteiro *n* e calcule o valor da série enquanto o termo for maior que 4/n. Execute os seguintes testes:

   abs (pi – seriePI 100) < 0.1
   abs (pi – seriePI 10000) < 0.001