**LISTA 1**

\*Nome da função + parâmetros = oq vai fazer

\* botar p salvar automático

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

ehTriangulo a b c = **if** a < b + c &&

b < c + a &&

c < b + a **then True else False**

\*\* && = e (and)

1. **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"

Equilátero -> a == b && b ==c

Isósceles -> a =/ b && b == c ou a == b e b = / c ou a == c e a=/ b

Escaleno -> a =/ b && b =/ c

tipoTriangulo a b c = if a==b && b==c then "equilatero" else if a /= b && b==c || a==b && b /= c || a==c && a/= b then "isosceles" else "escaleno"

QUANDO TEM MUITOS CASOS DA PRA FAZER ASSIM:

TipoTriangulo a b c | a == b && b==c = “equilatero”

| a == b || b==c || a ==c = “isosceles”

otherwise = “escaleno”

1. **Declare 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”, use as funções declaradas anteriormente.**

triangulo 5 5 5 => "equilatero"

triangulo 5 3 3 => "isosceles"

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

* ehTriangulo a b c = **if** a < b + c &&

b < c + a &&

c < b + a **then True else False**

* tipoTriangulo a b c = if a==b && b==c then "equilatero" else if a /= b && b==c || a==b && b /= c || a==c && a/= b then "isosceles" else "escaleno"

f3 l1 l2 l3 = if ehTriangulo l1 l2 l3 then tipoTriangulo l1 l2 l3 else "n eh triangulo"

1. **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

Somapares n =

Soma pares 1 = 0

Soma pares 2 = 0 + 2

Soma pares 3 = 0 + 2

Soma pares 4 = 4 + 2 + 0

somapares 1 = 0

somapares n = if mod n 2 == 0 then somapares (n-1) + n else somapares (n-1)

**5. Declare uma função que receba inteiros (m e n) e retorne a seguinte série: 2 0m + 2 1m + 2 2m+ ... + 2 nm. Por exemplo: somaPot2m 6 4 => 6 + 12 + 24 + 48 + 96 = 186 somaPot2m 3 3 => 3 + 6 + 12 + 24 = 45**

-- questão 5

somapot2m m 0 = m

somapot2m m n = (2^n) \* m + somapot2m m (n - 1)

**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 serie enquanto o termo for maior que 4/n. Execute os seguintes testes:**

abs (pi – seriePI 100) < 0.1

abs (pi – seriePI 10000) < 0.001