|  |  |  |
| --- | --- | --- |
| ‘m-logo1 | **Universidade Presbiteriana Mackenzie**  **- Faculdade de Computação e Informática –**  **Ciência da Computação**  **Paradigmas de Linguagens de Programação – 05N**  **Prova 2 – 19 de junho de 2020**  **Professor: Fabio Lubacheski** |  |

Esta prova pode ser feita em dupla ou individualmente, basta que somente um dos integrantes entregue um **arquivo pdf** com as repostas das questões abaixo, o arquivo deve conter oseguinte cabeçalho no início do arquivo.

/\*

Nós,

Luan Rocha D’Amato - 31817051

declaramos que

todas as respostas são fruto de nosso próprio trabalho,

não copiamos respostas de colegas externos a dupla,

não disponibilizamos nossas respostas para colegas externos a dupla e

não realizamos quaisquer outras atividades desonestas para nos beneficiar ou prejudicar outros.

\*/

Importante:

As soluções dos exercícios devem estar implementados em **Haskell**.

**1)** **(1,0 pontos)** Escreva em Haskell as expressões lambda abaixo e apresente a chamada da expressão em Haskell usando valores válidos, apresente também o resultado do cálculo para os valores informados.

1. (λ x y.x + y)

lambdaSoma a b = (\x y-> x+y)a b



1. (λ x.x2)

lambdaQuadrado a = (\x-> x^2)a



1. (λ x.(λ y. x\*y))

lambdaY a b = (\x -> (\y -> x\*y)b)a



1. (λ x.(λ y. xy))

lambdaElevadoY a b = (\x -> (\y -> x\*\*y)b)a



**2)** **(2,0 pontos)** Escreva uma função que receba 3 valores quaisquer e verifique se os valores podem ser considerados uma tripla de Pitágoras, ou seja, a soma dos quadrados de dois números é igual ao quadrado terceiro. Caso tenhamos uma tripla de Pitágoras a função devolve “eh uma tripla de Pitagoras” e caso não seja o a função devolve

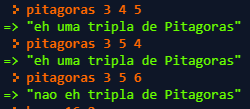
pitagoras l1 l2 l3

|l1\*\*2 + l2\*\*2 == l3\*\*2 = "eh uma tripla de Pitagoras"

|l2\*\*2 + l3\*\*2 == l1\*\*2 = "eh uma tripla de Pitagoras"

|l3\*\*2 + l1\*\*2 == l2\*\*2 = "eh uma tripla de Pitagoras"

|otherwise = "nao eh tripla de Pitagoras"



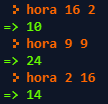
**3)** **(2,0 pontos)** Escreva uma função que recebe a hora inicial e a hora final de um jogo. A seguir a função calcula a duração do jogo, sabendo que o mesmo pode começar em um dia e terminar em outro, tendo uma duração mínima de 1 hora e máxima de 24 horas. Abaixo exemplos de valores informados para a função e o valor devolvido:

hora h1 h2

|h1 < h2 = h2-h1

|h1 == h2 = 24

|otherwise = (24-h1)+h2

****

**4)** **(2,0 pontos)** A ideia do algoritmo de **Multiplicação Russa** consiste em:

a. Escrever os números A e B, que se deseja multiplicar na parte superior das colunas.

b. Dividir A por 2, sucessivamente, ignorando o resto até chegar à unidade, escrever os resultados da coluna A.

c. Multiplicar B por 2 tantas vezes quantas se haja dividido A por 2, escrever os resultados sucessivos na coluna B.

d. Somar todos os números da coluna B que estejam ao lado de um número ímpar da coluna A.

Escreva uma **função recursiva** em Haskell que calcula a **Multiplicação Russa** de 2 entradas. Supondo que sua função tenha o nome de russa, a sua chamada deverá ser: russa 27 82 🡺 2214

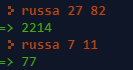
russa x y = aux x y 0

where aux x y soma

|x ==1 = (soma+y)

|mod x 2 == 0 = aux (div x 2) (y\*2) soma

|otherwise = aux (div x 2) (y\*2) (soma+y)



**5)** **(1,5 pontos)** Dada uma lista de inteiros, escreva uma função que devolve a quantidade de elementos **pares** na lista.

listaPares vetor = length([ x | x <- vetor, mod x 2 == 0])



**6)** **(1,5 pontos)** Dada uma lista de inteiros e um valor inteiro m, escreva uma função que devolve uma lista com todos os elementos menores ou igual a m. Nessa questão você deve usar um gerador para resolver o problema.

lista1 vetor m = [ x | x <- vetor, x <= m]



**Código Completo:**

--1a)

lambdaSoma a b = (\x y-> x+y)a b

--b)

lambdaQuadrado a = (\x-> x^2)a

--c)

lambdaY a b = (\x -> (\y -> x\*y)b)a

--d)

lambdaElevadoY a b = (\x -> (\y -> x\*\*y)b)a

--2)

pitagoras l1 l2 l3

|l1\*\*2 + l2\*\*2 == l3\*\*2 = "eh uma tripla de Pitagoras"

|l2\*\*2 + l3\*\*2 == l1\*\*2 = "eh uma tripla de Pitagoras"

|l3\*\*2 + l1\*\*2 == l2\*\*2 = "eh uma tripla de Pitagoras"

|otherwise = "nao eh tripla de Pitagoras"

--3)

hora h1 h2

|h1 < h2 = h2-h1

|h1 == h2 = 24

|otherwise = (24-h1)+h2

--4)

russa x y = aux x y 0

where aux x y soma

|x ==1 = (soma+y)

|mod x 2 == 0 = aux (div x 2) (y\*2) soma

|otherwise = aux (div x 2) (y\*2) (soma+y)

--5)

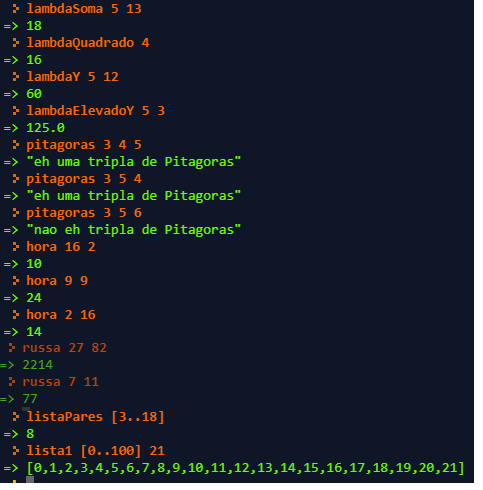
--listaParesImprime vetor = [ x | x <- vetor, mod x 2 == 0]

listaPares vetor = length([ x | x <- vetor, mod x 2 == 0])

--6)

lista1 vetor m = [ x | x <- vetor, x <= m]

Resoluções:



**Boa Prova !**