Universidade Presbiteriana Mackenzie  
Ciência da Computação – 05P11  
Paradigmas de Linguagens de Programação  
02/03/2025

Alan Meniuk Gleizer  
RA 10416804

Atividade – Paradigma Funcional

**01. Escreva um função que recebe um número e verifica se ele é par. A função retorna True caso o número seja par ou False caso contrário.**

**02. Usando a função par escreva a função impar que verifica se um número é impar utilizando a função par para calcular a resposta, o resultado da função par é negado com o operado not do Haskell.**

**03. Dado um n, escreva uma função que devolva a diferença absoluta entre n e 21, exceto devolva o dobro da diferença absoluta se n for maior que 21.**

**04. Escreva uma função que receba três valores para os lados de um triângulo. Na função verifique se os lados fornecidos realmente formam um triângulo, se formarem devolva tipo de triângulo temos: isósceles, escaleno ou equilátero.**

**05. 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 “nao eh tripla de Pitagoras”. Exemplos:**

**3 5 4 é uma tripla de Pitágoras**

**5 3 4 é uma tripla de Pitágoras**

**2 4 3 Não é tripla de Pitágoras**

**- // -**

**01. A sequência [0, 1, 1, 2, 3, 5, 8, 13, 21, ...] é conhecida como sequência ou série de Fibonacci, e tem aplicações teóricas e práticas, na medida em que alguns padrões na natureza parecem segui-la. Pode ser obtida através da recorrência abaixo: Escreva uma função em Haskell que calcula a sequência de Fibonacci.**

**Fib(𝑛) = 0 𝑠𝑒 𝑛 = 0**

**1 𝑠𝑒 𝑛 = 1**

**Fib(𝑛 − 1) + Fib(𝑛 − 2) 𝑠𝑒 𝑛 > 1**

**02. Imagine que a linguagem Haskell não possui mais o operador de multiplicação (\*), por sorte os matemáticos definiram que a multiplicação pode ser definida através de somas sucessivas:**

**a\*b = a + a ...+ a ou seja a somado b vezes.**

**Para a =4 e b =5 teremos 4+4+4+4+4, ou seja, 4 somado 5 vezes.**

**Escreva a função recursiva em Haskell que tem como entrada a e b, a função calcula e devolve a multiplicação de a\*b utilizando a regra acima.**

**03. Imagine agora que a linguagem Haskell não possui o operador de potência (^ ou \*\*), sabemos que a potência pode ser definida através de multiplicações, assim:**

**x^y = x \* x ...\*x ou seja x multiplicado y vezes.**

**Para x=2 e y=5 teremos 2\*2\*2\*2\*2 que é igual a 32.**

**Escreva a função recursiva em Haskell que tem como entrada x e y, a função calcula e devolve a potência de x^y utilizando a regra acima.**

**04. Imagine agora que a linguagem Haskell não possui o operador de potência (^ ou \*\*), nem o operador de multiplicação (\*), sabemos que a potência pode ser definida através de multiplicações e a multiplicação pode ser definida por somas sucessivas, assim:**

**x^y = x \* x ...\*x ou seja x multiplicado y vezes.**

**Para x=2 e y=5 teremos 2\*2\*2\*2\*2 que é igual a 32. Escreva a função recursiva em Haskell que tem como entrada x e y, a função calcula e devolve a potência de x^y utilizando a regra acima, para calcular a multiplicação utilize a função implementada no exercício 2.**

**05. A ideia do algoritmo de Multiplicação Russa consiste em:**

**a. Escreve em um papel os números A e B, que se deseja multiplicar na parte superior das colunas.**

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

**c. Multiplicar B por 2 tantas vezes quantas se tenha 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.**

**Exemplo: 27×82 = 2214**

**A B Parcelas**

**27 82 82**

**13 164 164**

**6 328 -**

**3 656 656**

**1 1312 1312**

**Soma: 2214**

**Escreva uma função em Haskell que calcula a multiplicação russa de 2 entradas;**

**06. A função abaixo calcula o máximo divisor comum dos inteiros positivos m e n usando o algoritmo de Euclides. Escreva uma função em Haskell que faz o mesmo cálculo.**

**int mdc( int m, int n ){**

**while( n != 0){**

**r = m % n;**

**m = n;**

**n = r**

**}**

**return m;**

**}**

**07. A expressão abaixo converge para a raiz quadrada de A, sendo A>0. Para começar, tente entender o funcionamento do algoritmo abaixo para o cálculo da raiz quadrada de A=2 com 5 iterações. Sugestão: faça um teste de mesa. Entendido o algoritmo escreva uma função em Haskell que calcule a raiz quadrada de um número usando o algoritmo definido pela expressão acima, considere que sua função terá o número de “iterações” (chamadas recursivas) informado por parâmetro.**

**08. O fatorial duplo de um número natural n é o produto de todos os números de 1 (ou 2) até n, contados de 2 em 2. Por exemplo, o fatorial duplo de 8 é 8x6x4x2 = 384, e o fatorial duplo de 7 é 7x5x3x1 = 105. Defina uma função em Haskell para calcular o fatorial duplo usando recursividade.**

**09. A raiz quadrada inteira de um número inteiro positivo n é o maior número inteiro cujo quadrado é menor ou igual a n. Por exemplo, a raiz quadrada inteira de 15 é 3, e a raiz quadrada inteira de 16 é 4. Defina uma função recursiva em Haskell para calcular a raiz quadrada inteira.**

**10. Escreva uma função equivalente em Haskell para o código a função abaixo que está no paradigma imperativo:**

**int função( int n) {**

**int x = 1;**

**for (int i = 1; i <= 10; i++) {**

**x = x\*2;**

**}**

**return x;**

**}**