Universidade Federal de Uberlandia - UFU
Faculdade de Computação - FACOM
Lista de Exercícios de Programação Funcional

Dica: Resolva todos os exercícios no papel e, somente quando estiverem prontos, implemente-os no computador.

1. Implemente o cálculo do mínimo múltiplo comum (MMC) de dois números:
Main > mmc (2, 3)
6
Utilize, para isso, a função mdc vista em aula. Sabe-se que:

$$mmc(a,b)=\frac{a.b}{mdc(a,b)}$$

2. Pesquise e implemente um metodo recursivo para calcular o MDC de três números. Utilize a função `mdc` vista em aula. Sabe-se que:

$$mdc(a,b,c)=mdc(mdc(a,b),c)$$

3. Implemente uma função recursiva `soma n :: Int -> Int` que computa a soma dos números de 1 a n.

4. Implemente uma função recursiva para calcular a soma entre dois números `n1` e `n2` incluindo os limites. Em seguida reimplemente essa função para excluir os limites.

5. Implemente uma função recursiva que, dados dois números `n1` e `n2`, encontra os multiplos de um terceiro numero `n3` que se encontram nesse intervalo.

6. Implemente uma função recursiva que calcule o número de grupos distintos com k pessoas que podem ser formados a partir de um conjunto de n pessoas (ou seja, a combinação de `n` pessoas em grupos de `k`. A definição abaixo da função `comb(n, k)` define as regras:

$$comb(n,k)=n \quad se \quad k=1$$
$$comb(n,k)=1 \quad se \quad k=n$$
$$comb(n,k)=comb(n-1,k-1)+comb(n-1,k) \quad se \quad 1<k<n$$

7. Seja a função e^x definida pela seguinte serie de Taylor:

$$e^x=1+\frac{x^1}{1!}+\frac{x^2}{2!}+\frac{x^3}{3!}+\ldots+\frac{x^n}{n!}$$

Implemente o calculo recursivo da soma da série para n = 10 termos e teste para varios valores de x. Compare os resultados obtidos com o valor dado pela função `exp x` do preludio-padrão.

8. Escreva uma função recursiva `conta_digitos` que recebe um numero inteiro n e retorna sua quantidade de dígitos. Exemplo: se n = 132, `conta_digitos n` retorna 3.

9. Escreva uma função recursiva `soma_digitos` que recebe um numero inteiro n e retorna a soma de seus dígitos. Exemplo: se n = 132, `soma_digitos n` retorna 6.

10. Implemente a função recursiva `potencia (b, e) :: (Int, Int) -> Int` que eleva a base b ao expoente e.

11. Implemente a função de Ackermann, a qual é definida por:

$$A(m,n) = n+1 \quad se \quad m=0$$
$$A(m,n) = A(m-1,1) \quad se \quad m>0 \quad e \quad n=0$$
$$A(m,n) = A(m,n-1) \quad se \quad m>0 \quad e \quad n>0$$

Observação: Teste essa função com valores pequenos (em torno de 0 a 3).

12. Desconsidere o conhecimento da função sqrt na linguagem Haskell. Uma forma de se obter a raiz quadrada de um numero qualquer x seria através de busca binária:
• Assuma que a raiz quadrada de x esta entre 1 (início) e x (fim), se x ≥ 1;
• Assuma que a raiz quadrada de x esta entre x (início) e 1 (fim), se x < 1;
• Se o número for negativo, retorne 0.

Para sabermos se um palpite `y = (inicio+fim)/2` é a raiz quadrada de x, basta testar se `y * y` é próximo o suficiente de x ou, em outras palavras, se o módulo da diferença entre x e `y * y` está dentro de uma tolerância definida. Caso contrário, podemos restringir a busca entre início e y ou entre y e fim. Escreva a função que implemente este algoritmo, considerando 10^{-6} como tolerância para o cálculo do resultado.