Engenharia de Software
Campus Garanhuns

Disciplina: Programação 3 - Programação Funcional
Professor: Emanoel Barreiros
Assunto: Introdução, listas, definição de funções.

Resumo: Utilize a linguagem de programação Haskell para resolver os problemas desta lista.

1) Fornecidos três valores *a, b* e *c*, escreva uma função que retorne quantos dos três são iguais. A resposta pode ser 3 (todos iguais), 2 (dois iguais e o terceiro diferente) ou 0 (todos diferentes).

2) Fornecidos três valores *a, b* e *c*, elaborar uma função que retorne quantos desses três valores são maiores que a média entre eles.

3) Escreva uma função `potencia_2` que retorne o quadrado de um número (x^2).

4) Reutilizando a função `potencia_2`, construir uma função `potencia_4` que retorne o seu argumento elevado à quarta potência.

5) Implemente em Haskell a função do ou-exclusivo, que é dada por:

$$a \otimes b = (a \vee b) \wedge \neg (a \wedge b)$$

6) Escrever duas funções, x_maior que retorne o maior e x_menor que retorne o menor valor real, das raízes de uma equação do segundo grau. A expressão genérica é dada por:

$$\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

7) Criar funções que calculem a soma dos números entre *n1* e *n2*, incluindo e excluindo os limites.

8) Dados dois números *n1* e *n2*, encontrar os múltiplos de *n3* que se encontram nesse intervalo (inclusivo).

9) Utilizando a função soma, faça uma função que calcule a multiplicação entre dois números quaisquer, considerando números positivos e negativos.

10) Implemente a função `mod2`, que retorna o resto de uma divisão de inteiros. OBS: não é permitido usar a função `mod` da biblioteca.

11) Seja a sequência:

$$a_1 = \sqrt{6}$$

$a_2 = \sqrt{6 + \sqrt{6}}$

$a_3 = \sqrt{6 + \sqrt{6 + \sqrt{6}}}$

a_4 =...

Escreva a função que calcula essa sequência.

12) Implementar a fórmula que indica de quantas maneiras é possível escolher *n* objetos de uma coleção original de *m* objetos, onde *m* >= *n*.

13) Defina uma função que, dada uma lista numérica, retorne uma tupla, que contenha o maior da lista bem como sua posição relativa.

14) Defina uma função que converta uma lista de dígitos (unitários, 0 a 9) e uma outra lista, que é a sua tradução em String. Considere um dicionário do tipo:

dic_10 = [(0,"zero"), (1,"um"), (2,"dois"), …, (9,"nove")]

15) Construa uma função `del_posicao_n :: [Int] -> Int -> [Int]` em que dada uma lista de inteiros e a posição de um elemento qualquer, retorne uma nova lista sem aquele elemento da n-ésima posição.

16) Construa uma função inserir_posicao_x :: [Int] -> Int -> Int -> [Int] em que, dada uma lista de inteiros, uma posição e um elemento a ser inserido, retorne uma nova lista com aquele elemento na n-ésima posição.

17) Defina uma função que retorne o valor da n-ésima posição de uma lista.

18) Dadas duas listas ordenadas como entrada, faça uma função `merge`, que une as duas listas.

19) Implemente uma função que receba duas listas de inteiros sem repetição, e retorne uma terceira lista que contenha somente números que estejam nas duas listas dadas.

20) Crie uma função que faça uma codificação sobre uma sequência de caracteres iguais, substitua a sequência por **!na**, onde **n** é o número de vezes que o caractere **a** é repetido. Observe que só devem ser comprimidas sequências de tamanhos maiores que 3. Exemplo:

comprime "asdfffghjjkllllpoooi"
"asd!4fghjjk!4lpoooi"

21) Implemente uma função que descomprima o texto resultante da função anterior.