Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada
Linguagens de Programação: Conceitos e Paradigmas

Programação em Haskell - Exercícios 01

Aluno (a):_____

1. Apresente uma definição para uma função semZeroNoPeriodo::Int->Bool, que retorna True somente se não há nenhum dia no intervalo 0,1,...,n em que o número de vendas foi zero. Inspire-se nas funções discutidas em sala de aula, mas crie sua função a partir do zero, sem utilizar a função ZeroNoPeriodo.

2. Escreva uma definição equivalente à exibida abaixo, mas usando apenas uma única cláusula simples.

```
funct  x y z
  | x > z        = True
  | y >= x       = False
  | otherwise = True
```

3. Defina uma função que converte letras minúsculas para maiúsculas, mas que retorna o mesmo caractere fornecido como entrada, se este não for uma letra minúscula.

   Dica: a função pré-definida isLower::Char->Bool retorna True se o caractere fornecido for uma letra minúscula.

4. Defina um operador binário de nome &-, onde: x &- y = x - 2*y.

   Qual o resultado da avaliação da expressão 10 &- 5 &- 2 , se o operador for definido como infixl 6 &-, como infixr 6 &- e como infix 6 &- ?

   Qual o resultado da avaliação da expressão 10 &- 3 * 2, se o operador for definido como infix 6 &-, e como infix 8 &- ?

   Explique esses resultados.

5. Construa uma função que encontre o número do dia em que o maior número de vendas ocorreu num determinado período (dia 0 até dia n). Como sua função se comporta no caso em que o máximo valor acontece em mais de um dia? Com base em sua resposta, apresente uma especificação mais precisa para a função.

6. Defina uma função que calcule o número de dias em que as vendas foram inferiores a um dado valor, dentro de um dado período:

   howManyLess :: Int -> Int -> Int

   onde o primeiro parâmetro indica o valor mínimo, e o segundo parâmetro indica o último dia do período (dia 0 ao dia n).

7. A seqüência 0, 1, 1, 2, 3, 5, 8, 13, 21, ... é conhecida como Seqüência de Fibonacci. Os dois primeiros números são 0 e 1, e os demais são sempre a soma dos dois anteriores. Construa uma função fib :: Integer -> Integer que calcule um número de Fibonacci, dada sua posição na seqüência. Uma solução ingênua foi apresentada em sala de aula:

   ```
   fib 0      = 0
   fib 1      = 1
   fib n      =  fib (n - 2) + fib (n - 1)
   ```

   Essa solução é muito ineficiente, pois pode executar várias chamadas repetidas para um mesmo valor de entrada. Por exemplo, fib 5 executa fib 2 três vezes. Como sugestão para uma solução mais eficiente, crie uma função

   fibDupla :: Integer -> (Integer,Integer)

   que, dado um número n > 0, retorna o par (a,b), onde b é o número de Fibonacci da posição n e a é o número de Fibonacci da posição n-1. Esboço de solução:

   ```
   fib 0 = 0
   fib n =  segundo elemento do par retornado por fibDupla n ...

   fibDupla 1 = (0,1)
   fibDupla n =  usar fibDupla (n-1) e recursividade ...
   ```