IF686 - 2020.3
Lista de exercícios
Listas

1. Defina a função

   paraMaiuscula :: String -> String

   que transforma toda letra minúscula em uma string em maiúscula. Use compreensão de listas.

   Modifique esta função para se comportar da mesma forma, mas removendo todos os caracteres que não são letras da string resultante.

2. Defina a função

   divisores :: Integer -> [Integer]

   que retorna uma lista de divisores e um inteiro positivo e uma lista vazia para outras entradas

   Use divisores para definir a função

   isPrime :: Integer -> Bool

   que testa se um número é primo.

3. Dada uma lista de inteiros, defina a função menorLista que encontra o menor inteiro dessa lista

4. Defina a função

   fibTable :: Integer -> String

   que produz uma tabela com os números de Fibonacci. A saída para fibTable 6 é

n	fib n
0	0
1	1
2	1
3	2
4	3

5. Defina função measure que, para um lista vazia, retorna -1 e ,para outras listas, retorna o tamanho da lista

6. Defina a função takeFinal que retorna os últimos n elementos de uma lista dada como argumento

7. Defina uma função que remove o enésimo elemento de uma lista, ou seja, retorna uma lista que idêntica à lista recebida como argumento, com exceção de que o enésimo não consta na lista de retorno. A indexação começa em zero. Exemplos:

```
remove 0 [1,2,3]     ==>  [2,3]
remove 2 [4,5,6,7]   ==>  [4,5,7]
```

8. Dê uma definição com casamento de padrão de uma função que retorna o primeiro inteiro, se houver, em uma lista incrementado de um, ou retorna zero, do contrário. Apresente também uma solução sem casamento de padrão, mas usando funções da biblioteca.

9. Dê uma definição com casamento de padrão de uma função que faz a adição dos dois primeiros elementos de uma lista, se a lista contém pelo menos dois elementos; retorna a cabeça da lista, se houver um elemento apenas; retorna zero, do contrário. Apresente também uma solução sem casamento de padrão, mas usando funções da biblioteca.

10. Defina a função

    ```
    produto :: [Integer] -> Integer
    ```

    que retorna o produto de uma lista de inteiros e 1, no caso da lista vazia.

11. Defina a função

    ```
    unique :: [Integer] -> [Integer]
    ```

    que retorna os elementos que ocorrem apenas uma vez na lista dada como argumento.

12. Defina uma função que verifica se uma lista dada como argumento está em ordem crescente. Use recursão e casamento de padrão. Não use funções de bibliotecas