```
import Data.List
import Data.Char
-- Gabriel Ribeiro Bernardi - 11821BCC036

-- Questao 01

-- considerando que a entrada sera uma string
-- éPalíndromo :: String -> Bool
-- éPalíndromo s
--     | s == reverse s = True
--     | otherwise = False

-- funcao polimorfica
éPalíndromo :: Eq a => [a] -> Bool
éPalíndromo s
    | s == reverse s = True
    | otherwise = False

-- Questao 02

-- éPangrama "abcdefghijklmnopqrstuvwxy"
-- False

-- éPangrama "abcdefghijklmnopqrstuvwxyz"
-- True

-- éPangrama "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
-- True

-- éPangrama "Bancos fúteis pagavam-lhe queijo, whisky e xadrez"
-- True

-- éPangrama "Já fiz vinho com toque de kiwi para belga sexy"
-- True

-- éPangrama "The quick brown fox jumps over the lazy dog"
-- True

-- éPangrama "Huguinho, Zézinho e Luisinho"
-- False

converteParaMinusculo :: String -> String
converteParaMinusculo entrada = map toLower entrada

defineAlfabeto :: String
defineAlfabeto = "abcdefghijklmnopqrstuvwxyz"

verificaString :: String -> Bool
verificaString entrada = all (`elem` converteParaMinusculo entrada )
defineAlfabeto

éPangrama :: String -> Bool
éPangrama entrada = verificaString entrada
```

```
--Questao 03
--1
data ArvoreBinaria0 a = Nulo0 | No0 a (ArvoreBinaria0 a) (ArvoreBinaria0 a)
deriving Show

-- 2
arvExemplo = (No0 "a" (No0 "b" (No0 "d" Nulo0 Nulo0) Nulo0)
                      (No0 "c" (No0 "e" (No0 "g" Nulo0 Nulo0) Nulo0)
                                        (No0 "f" Nulo0 Nulo0)))

-- naOrdem :: ArvoreBinaria [Char] -> [[Char]]
-- naOrdem Nulo = []
-- naOrdem (No n esq dir) = naOrdem esq ++ [n] ++ naOrdem dir

--Questao 04

aplicação f x = f $ x

xAplicação f x = f $ (aplicação f x)

xXAplicação f x n
    | n == 0 = x
    | otherwise = xXAplicação f (aplicação f x) (n-1)

--Questao 05
-- média [1,2,3]
-- 2.0

-- média []
-- *** Exception: nao existe media de lista vazia

-- média [-3,-2,-1,0,1,2,3,4]
-- 0.5

somaLista [] = 0
somaLista (cabeca:rabo) = cabeca + somaLista rabo

-- foi necessario fazer a conversao dos valores no momento em que faria a divisao
média lista
    | null lista = error "nao existe media de lista vazia"
    | otherwise = fromIntegral (somaLista lista) / fromIntegral (length lista)

-- médias [[1,2,3],[4,5,6]]
-- [2.0,5.0]

-- médias [[1,2,3],[4,5,6],[1,2,3]]
-- [2.0,5.0,2.0]

-- médias [[1,2,3]]
-- [2.0]

médias lista
    | null lista = []
    | otherwise = média (head lista) : médias (tail lista)
```

```
--Questao 06

data ArvoreBinaria a = Nulo | No a (ArvoreBinaria a) (ArvoreBinaria a) deriving
Show

arvEx = (No "cansado" (No "sim" (No "eh noite" (No "sim" (No "dormir" Nulo Nulo)
Nulo)
                                                (No "nao" (No "trabalhando" (No
"sim" (No "complicou" Nulo Nulo) Nulo)
                                                                            (No
"nao" (No "cochilar" Nulo Nulo) Nulo) ) Nulo ) ) Nulo)
                      (No "nao" (No "trabalhando" (No "sim" (No "promocao" Nulo
Nulo) Nulo)
                                                  (No "nao" (No "va correr" Nulo
Nulo) Nulo)) Nulo))

-- Questao 07
-- contaValores [[1,1],[12],[2,2],[3,3,3],[4,4,4],[3,3,3]]
-- [(1,2),(12,1),(2,2),(3,3),(4,3),(3,3)]

contaValores :: [[a]] -> [(a, Int)]
contaValores l
    | null l = [] --verifica se lista eh nula
    | otherwise = (head (primeiro), length primeiro) : contaValores restante
    where
        primeiro = head l
        restante = tail l
```