
Data;Tipo;Compra;Valor;
14 JAN;Amazon;40.32;
15 JAN;Uber;14.84;
25 JAN;Uber;34.24;
02 FEV;Spotify;8.50;
06 FEV;Uber;6.94;
05 MAR;Burger;29.90;
10 MAR;Burger;24.99;
15 MAR;UCI;19.00;
08 ABR;Itunes;3.50;
13 ABR;Picpay;20.00;

Considere a fatura anual de cartão acima, sendo representada pela String:
logCartao = "14 JAN;Amazon;40.32;15 JAN;Uber;14.84;25 JAN;Uber;34.24;02 FEV;Spotify;8.50;06 FEV;Uber;6.94;05 MAR;Burger;29.90;10 MAR;Burger;24.99;15 MAR;UCI;19.00;08 ABR;Itunes;3.50;13 ABR;Picpay;20.00;"

[Q1] Escreva uma função **logMes :: String -> String -> Double** que recebe uma String (JAN, FEV, MAR ou ABR), uma String referente a fatura anual e retorna o total gasto no mês em questão.

```
Exemplo:
Main> logMes "JAN" logCartao
89.4
```

[Q2] Escreva uma função **minMaxCartao :: String -> (Double, Double)** que recebe uma String referente a fatura anual e retorna uma tupla com o menor e o maior dos valores gastos.

```
Exemplo:
Main> minMaxCartao logCartao
(3.5, 40.32)
```

[Q3] A string **"aaaaa"** repete um único caractere, o **'a'**, 5 vezes. Crie uma função **isReplica -> String -> Int -> Char -> Bool** que recebe uma string, um inteiro **x** e um char verifica se essa string é a repetição do char **x** vezes.

```
isReplica "ee" 2 'e'
True
isReplica "uruu" 3 'u'
False
isReplica "xxx" 3 'y'
```

False

[Q4] Crie uma função **decEnigma :: String -> [(Char, Char)] -> String** que decifra uma string da linguagem A para a linguagem B. Ela recebe a string que precisa ser decifrada e uma lista de tuplas contendo os dois alfabetos. Os primeiros caracteres da tupla, representam o alfabeto de A e, os segundos, de B.

decEnigma "usr" [('u','j'), ('s','o'), ('r','b')]
"job"
decEnigma "msyc" [('m','e'), ('s','i'), ('y','t'), ('c','a')]
"eita"
decEnigma "qloz" [('q','h'), ('l','u'), ('o','g'), ('z','o')]
"hugo"
decEnigma "usr" [('s','o'), ('u','j'), ('m', 't'), ('r','b')]

[Q5] Faça uma função em Haskell **btoi :: String -> Int** que, dada uma string representando um número binário, retorna um inteiro na base 10 dessa string.

btoi "0011"
3
btoi "1100"
12

[Q6] Crie uma função **executa :: [(Comando, Valor)] -> Int** que recebe uma lista de tuplas com comandos e valores. A função deve começar pelo valor. Por exemplo, caso a sequência de comandos seja [("Multiplica", 2), ("Soma", 5), ("Subtrai", 3)], a função deve pegar 0 e efetuar as seguintes operações: (((0 * 2) + 5) -3). Esses comandos podem ser "Multiplica", "Soma", "Subtrai" ou "Divide". Para o caso de uma divisão por 0, a função deve retornar o valor -666 independente de quanto tenha calculado até essa divisão.

```
type Comando = String
type Valor = Int

executa [("Multiplica", 2), ("Soma", 5), ("Subtrai", 3)]
2
executa [("Multiplica", 2)]
0
executa [("Multiplica", 2), ("Soma", 5), ("Subtrai", 3), ("Soma",
6)]
8
executa [("Multiplica", 2), ("Soma", 5), ("Divide", 0)]
-666
```

[Q7] Crie uma função **mul2 :: [Int] -> [Int] -> [Int]** que recebe 2 listas de inteiros e retorna uma lista com o produto elemento por elemento
entre as duas listas. Caso as listas não tenham o mesmo tamanho multiplique por 0 os elementos restantes.

```
mul2 [1,2,3] [3,3,3]
[3,6,9]

mul2 [1,2] [4,5,6]
[4,10,0]
```