Linguagens Declarativas
Primeira Lista de Exercícios (aquecimento)

Rodrigo Bonifácio

Problema 01

Nos bons e velhos tempos, era possível escrever artigos com títulos como *The morphology of prex— an essay in meta-algorithmics.* Atualmente, jornais parecem preferir todas as palavras em caixa-alta, como *The Morphology Of Prex— An Essay In Meta-algorithmics.* Escreva uma função `modernize :: String -> String` que garante que os títulos dos artigos fiquem em caixa-alta, como o exemplo acima. (fonte: Richard Bird. *Thinking Functionally with Haskell.* Cambridge Press, 2014).

Problema 02

Suponha que uma data seja representada por três inteiros `(day, month, year)`. Defina uma função `showDate :: Date -> String`, de tal forma que atenda aos seguintes exemplos:

```
showDate(10, 12, 2013) = "10th December, 2013"
showDate(21, 11, 2020) = "21st November, 2020"
```

(fonte: Richard Bird. *Thinking Functionally with Haskell.* Cambridge Press, 2014).

Problema 03

Crie um *password strenght checker*, considerando que uma senha robusta possui as seguintes características:

- pelo menos 10 caracteres
- pelo menos a ocorrência de uma letra maiúscula
- pelo menos a ocorrência de uma letra minúscula
- pelo menos a ocorrência de um número

Sugestões: as funções definidas nas bibliotecas `Data.List` e `Data.Char` podem ser úteis para a resolução desse problema. A função que verifica a robustez das senhas deve ter a seguinte assinatura: `strong :: String -> Bool`. Tente trabalhar com a notação de ponto-fixo, onde `g(f(x)) = (g . f) x`. Tal notação é mais legível e elegante; em particular quando consideramos algumas propriedades válidas, tais como `map g (map f xs) = map (g . f) xs`. Ou seja:

```
$ ghci
GHCi, version 7.8.3: http://www.haskell.org/ghc/  :? for help

Prelude> let inc = (+ 1) in map even (map inc [1..10])
[True,False,True,False,True,False,True,False,True,False]

Prelude> let inc = (+ 1) in map (even . inc) [1..10]
[True,False,True,False,True,False,True,False,True,False]
```

(fonte: Nishant Shukla. Haskell Lectures. on-line: http://shuklan.com/haskell/)

Problema 04

Defina um tipo algébrico para representar árvores binárias (`data Tree = ...`) que contêm nós de valores inteiros e escreva uma função `sumT :: Tree -> Int` que realiza uma traversia na árvore e calcula o somatório de todos os valores.
(fonte: Nishant Shukla. Haskell Lectures. on-line: http://shuklan.com/haskell/)