Programação Funcional
Folha de Exercícios 05
Entrada e Saída

Prof. João Marcelo

Entrada e Saída

1. Escreva uma função `seq_ :: [IO a] -> IO ()` que realiza uma lista de ações.

```
Prelude> seq_ [print i | i <- [1..5]]
1
2
3
4
5
```

2. Escreva uma função `elefantes :: Int -> IO ()` tal que, por exemplo, elefantes 5 imprime os seguintes versos:

```
Se 2 elefantes incomodam muita gente,
3 elefantes incomodam muito mais!
Se 3 elefantes incomodam muita gente,
4 elefantes incomodam muito mais!
Se 4 elefantes incomodam muita gente,
5 elefantes incomodam muito mais!
```

   Sugestão: utilize a função `show :: Show a => a -> String` para converter um inteiro numa cadeia de caracteres; pode ainda reutilizar a função `seq_ :: [IO a] -> IO ()` para executar uma lista de ações.

3. Considere o seguinte programa:

```
module Main where
main
        = do {
        tests <- getLine;
        contents <- getContents;
        putStrLn $ show $take (read tests) (lines contents)
        }
```

   Modifique o programa para que ele leia um número natural n, e então leia outros n números e calcule e exiba a soma destes números.

4. A função `interact :: (String -> String) -> IO ()` é muito utilizada para construir programas com entrada e saída simples. Considere o seguinte programa:

```
module Main where
main = interact (show.length.lines)
```

   O programa acima imprime o número de linhas do arquivo de entrada.

   Faça um programa completo que lê linhas de texto da entrada-padrão e imprime cada linha invertida usando a função interact.

   Dica: Use as funções lines, unlines, map reverse.

5. Escreva um programa completo que reproduza a funcionalidade do utilitário wc: ler um ficheiro de entrada e imprime o número de linhas, número de palavras e número de caracteres.

```
$echo a maria tinha um cordeirinho | wc
Linhas: 1
Palavras: 5
Caracteres: 29
```

   Sugestão: Utilize as funções `words :: String -> [String]` e `lines :: String -> [String]`.

6. Faça um programa que leia um número n e imprime n!
7. Faça um programa que leia um número n e imprime “sim”, se o número é primo e “nao”, caso contrário.
8. Escreva a função `accumulate::[IO a] -> IO[a]`, que realiza uma lista de ações, acumulando o resultado dessas ações em uma lista.
9. Faça programa para que ele leia um número natural n, e então leia outros n números e calcule e exiba a soma destes números usando a função `accumulate`.