Universidade Federal de Uberlândia - UFU
Faculdade de Computação - FACOM
Lista de exercícios de Programação Funcional
Ref: Funções de Ordem Superior

Dica: Resolva todos os exercícios sem utilizar o computador. Uma vez que os exercícios estejam prontos, utilize o GHCi para conferir suas respostas.

1. Nos ítens a seguir, escreva primeiramente uma função recursiva. Em seguida, reescreva essa função utilizando a função de ordem superior map, vista em aula.

(a) Função primeiros :: [(a,b)] -> [a] que extrai o primeiro elemento de cada tupla-2 dentro de uma lista. Dica: utilize a função fst

(b) Função maiusculas :: String -> String que converte uma string para outra string com letras maiusculas. Dica: utilize a função toUpper da biblioteca Data.Char.

(c) Função dobros :: Num a => [a] -> [a] que dobra todos os elementos de uma lista.

(d) Função hora_em_seg :: [Float] -> [Float] que converte uma lista de horas em uma lista de segundos.

2. Escreva uma função em Haskell usando map chamada ellen que considera uma lista de strings e devolve uma lista dos tamanhos de cada string

3. Reescreva a função do item anterior usando recursão.

4. Escreva uma função map2 que considera uma lista de funções e uma lista de valores (ambas do mesmo tamanho) e devolve a lista resultante da aplicação de cada função da primeira lista sobre o valor correspondente na segunda lista.

5. Escreva uma função fmap que considera um valor e uma lista de funções e devolve uma lista de resultados da aplicação de cada função sobre tal valor. Por exemplo: fmap 3 [((*)2), ((+)2)] resulta em [6,5].

6. Qual resultado da execução destas funções? Faça no papel e compare o resultado com a execução no GHCI.

```
Main > map sqrt [0 ,1 ,4 ,9]
Main > map succ " HAL "
Main > map head [" bom "," dia "," turma "]
Main > map even [8 ,10 , -3 ,48 ,5]
Main > map isDigit " A18 B7"
Main > map length [" ci^enca ", "da", " computa¸c~ao "]
Main > map ( sqrt . abs . snd ) [('A ' ,100) ,('Z ' , -36)]
```

7. Nos itens a seguir, escreva primeiramente uma função recursiva. Em seguida, reescreva essa função utilizando a funçãoo de ordem superior filter, vista em aula.

(a) Função pares :: [Int] -> [Int] que remove todos os elementos ımpares de uma lista. Dica: implemente uma função que verifique se um número é par.

(b) Função alfa :: String -> String que remove todos os caracteres nao-alfabéticos de uma string. Dica: utilize a função isAlpha da biblioteca Data.Char.

(c) Defina a função rm char :: Char -> String -> String que remove todas as ocorrencias de um caractere em uma string.

(d) Defina a função acima :: Int -> [Int] -> [Int] que remove todos os numeros menores ou iguais a um determinado valor.

(e) Escreva uma função desiguais :: Eq t => [(t,t)] -> [(t,t)] que remove todos os pares (x,y) em que x == y.

8. Nos exercícios a seguir, escreva primeiramente uma func¸ao recursiva. Em seguida, reescreva essa função utilizando a função de ordem superior foldr, vista em aula.

(a) Função produto :: Num a => [a] -> a que computa o produto dos numeros de uma lista.

(b) Função e_logico :: [Bool] -> Bool que “e logico” de todos os itens em uma lista.

(c) Função concatena :: [String] -> String que junta uma lista de strings em uma unica string.

(d) Função maior :: Num a => a -> [a] -> a que considera um numero e o compara com os elementos de uma lista. Caso esse numero seja maior que todos os elementos da lista, ele será retornado pela função. Caso contrário, a função retorna o maior elemento da lista. Por exemplo:
Main *> maior 18 [3 ,6 ,12 ,4 ,55 ,11]
55
Main *> maior 111 [3 ,6 ,12 ,4 ,55 ,11]
111
9. Para cada uma das seguintes especificações, escreva uma função em Haskell. Use as funções map, filter e foldr
(a) Função numof que considera um valor e uma lista e devolve o numero de ocorrências do valor na lista
(b) Função ellen que considera uma lista de strings e devolve uma lista dos tamanhos de cada string
(c) Função ssp que considera uma lista de inteiros e devolve a soma dos quadrados dos elementos positivos da lista
10. Defina a função sqp (soma dos quadrados dos primos) que considera uma lista de inteiros como argumento, remove os que nao são primos, eleva os inteiros restantes ao quadrado e, finalmente, soma tais quadrados. Por exemplo:
*Main > sqp [2, 4, 7, 1, 3]
62
11. Seja uma lista de strings. Utilizando funções de ordem superior, faça as seguintes funções:
(a) Uma funçãao que devolva o valor da soma dos comprimentos de cada string (elemento) da lista. Isto e, a soma total dos comprimentos da lista de entrada;
(b) Uma função que devolva uma lista das strings formadas por palavras que se iniciem por uma letras específicas;
(c) Uma função que devolva a mesma lista de entrada, exceto que as letras em cada palavra ficarão invertidas entre maiusculas e minusculas. Por exemplo, onde tem ‘e’ ´troque por ‘E’, onde tem ‘E’ troque por ‘e’, e assim por diante.
12. As seguintes funçõoes do prelúdio-padrão são de ordem superior. Mostre o resultado das seguintes execuções:
Main > all (<10) [1 ,3 ,5 ,7 ,9]
Main > all (==1) [1 ,1 ,0 ,1 ,1]
Main > all even [2 ,4 ,6 ,8 ,10]
Main > any (1==) [0 ,1 ,2 ,3 ,4 ,5]
Main > any (>5) [0 ,1 ,2 ,3 ,4 ,5]
Main > any even [1 ,3 ,5 ,7 ,9]
Main > takeWhile (<3) [1 ,2 ,3 ,4 ,5]
Main > takeWhile (>3) [1 ,2 ,3 ,4 ,5]
Main > takeWhile odd [1 ,3 ,5 ,7 ,9 ,10 ,11 ,13 ,15 ,17]
Main > takeWhile (’w ’>) " hello world "
Main > dropWhile (<3) [1 ,2 ,3 ,4 ,5]
Main > dropWhile even [2 ,4 ,6 ,7 ,9 ,11 ,12 ,13 ,14]
Main > dropWhile (’w ’>) "hello , world "