Programação Funcional - Lista de Exercícios 1

Luiz Alberto do Carmo Viana

8 de abril de 2018

Questão 1
Implemente a função `uncons :: [a] -> Maybe (a, [a])` que, dada uma lista, retorna uma tupla composta por sua cabeça e sua cauda. Note que a função `uncons` evita lançar uma exceção caso receba uma lista vazia.

Questão 2
Implemente a função `iterate :: (a -> a) -> a -> [a]` que, dados uma função f e um elemento x, cria uma lista infinita com os resultados de sucessivas aplicações de f em x, isto é, `iterate f x = [x, f x, f (f x), f (f (f x)), ...]`.

Questão 3
Implemente a função `replicate :: Int -> a -> [a]` que, dado um inteiro não-negativo n e um elemento x, cria uma lista de comprimento n tal que cada uma de suas posições contém o elemento x.

Questão 4
Implemente a função `group :: Eq a => [a] -> [[a]]` que, dada uma lista de entrada, retorna uma lista de listas, onde cada sublista contém elementos consecutivos iguais da lista de entrada. Exemplo: `group [1, 2, 2, 3, 3, 3, 1, 2, 3] == [[1], [2, 2], [3, 3, 3], [1], [2], [3]]`.

Questão 5
Implemente a função `inits :: [a] -> [[a]]`, que retorna todos os prefixos de uma lista dada como entrada.

Questão 6
Implemente a função `tails :: [a] -> [[a]]`, que retorna todos os sufixos de uma lista dada como entrada.

Questão 7
A função `isInfixOf :: (Eq a) => [a] -> [a] -> Bool` decide se a primeira lista ocorre como sublista **contínua** na segunda lista. Existe também a função `isSubsequenceOf :: (Eq a) => [a] -> [a] -> Bool`, que decide

se a primeira lista é uma subsequência da segunda, **não necessariamente contínua**. Implemente a função `isSubsequenceOf`.

Questão 8

Implemente a função `find :: (a -> Bool) -> [a] -> Maybe a`, que possivelmente retorna o primeiro elemento da lista de entrada que satisfaz o predicado.

Questão 9

A função `insert :: (Ord a) => a -> [a] -> [a]` insere um elemento de um tipo ordenável em uma lista ordenada, retornando uma lista ordenada. Implemente `insert` e use-a para implementar uma função `insertionSort :: (Ord a) => [a] -> [a]`, que toma uma lista de elementos ordenáveis e retorna uma versão ordenada dessa lista.

Questão 10

Defina as seguintes funções em termos de `foldr`. Você poderá usar as funções `(:) :: a -> [a] -> [a]`, `(&&) :: Bool -> Bool -> Bool`, `(||) :: Bool -> Bool -> Bool` e `(.) :: (b -> c) -> (a -> b) -> a -> c`.

- `id :: [a] -> [a]`, onde `id xs == xs`.
- `and :: [Bool] -> Bool`, que retorna `True` $\iff$ a lista contém apenas elementos `True`.
- `or :: [Bool] -> Bool`, que retorna `True` $\iff$ a lista contém algum elemento `True`.
- `map :: (a -> b) -> [a] -> [b]`, que nos é bastante conhecida.