Paradigmas de Programação

Fabrício Olivetti de França

05 de Julho de 2018

Lista de exercícios 04

1. Sem olhar as definições do Prelude padrão, defina as seguintes funções de alta-ordem:

```haskell
-- Verifica se todos elementos da lista satisfazem um predicado
all' :: (a -> Bool) -> [a] -> Bool

-- Verifica se pelo menos um dos elementos da lista satisfazem um predicado
any' :: (a -> Bool) -> [a] -> Bool

-- Seleciona os elementos da lista enquanto eles satisfizerem um predicado
takeWhile' :: (a -> Bool) -> [a] -> [a]

-- Remove os elementos da lista enquanto eles satisfizerem um predicado
dropWhile' :: (a -> Bool) -> [a] -> [a]
```

2. Defina uma função

```haskell
altMap :: (a -> b) -> (a -> b) -> [a] -> [b]
```

que aplica alternadamente as duas funções recebidas como argumento em uma lista. Exemplo:

```
> altMap (+10) (+100) [0,1,2,3,4]
[10,101,12,103,14]
```

3. Usando *folding*, defina a função

```haskell
dec2int :: [Int] -> Int
```

que converte uma lista de inteiros em um inteiro. Exemplo:

```
> dec2int [2,3,4,5]
2345
```

4. Utilize *folding* para definir as seguintes funções em Haskell:

```
-- retorna True se pelo menos um booleano da lista for True
or' :: [Bool] -> Bool

-- inverte os elementos de uma lista
reverse' :: [a] -> [a]

-- filtra os elementos de uma lista de acordo com um predicado
filter' :: (a -> Bool) -> [a] -> [a]
```

5. Escreva duas versões para a função

```
elem' :: Eq a => a -> [a] -> Bool
```

que verifica se um elemento está em uma lista. Uma versão deverá utilizar *folding* e outra a função *any*.