

Desafios de Listas

prof. André Rauber Du Bois

Universidade Federal de Pelotas
dubois@inf.ufpel.edu.br

1 Questionário

1. Implementar a função `slice`, que devolve um pedaço interno de uma lista:

```
*Main> slice 3 7 ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'k']
"cddefg"
```

2. Implementar a função `compress` que elimina elementos consecutivos

```
> compress "aaaabccaaadeeee"
"abcade"
```

3. Implementar a função `pack` que coloca elementos consecutivos em sublistas

```
> pack ['a', 'a', 'a', 'a', 'b', 'c', 'c', 'a',
         'a', 'd', 'e', 'e', 'e', 'e']
["aaaa", "b", "cc", "aa", "d", "eeee"]
```

```
> pack "bbaccczwwwq"
["bb", "a", "ccc", "z", "www", "q"]
```

4. Implementar a função `encode`:

```
> encode "aaaabccaaadeeee"
[(4, 'a'), (1, 'b'), (2, 'c'), (2, 'a'), (1, 'd'), (4, 'e')]
```

5. Implementar a função `rotate`

```
*Main> rotate ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'] 3
"defghabc"
```

```
*Main> rotate ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'] (-2)
"ghabcdef"
```

6. Em um exercício anterior, implementamos a função de ordenação `quickSort`:

```
quickSort :: [Int] -> [Int]
quickSort [] = []
quickSort (x:xs) = quickSort (menores x xs)
                  ++ [x] ++
                  quickSort (maiores x xs)
```

onde o objetivo era implementar as funções:

```
menores :: Int -> [Int] -> [Int]
maiores :: Int -> [Int] -> [Int]
```

sendo que `menores` recebe um inteiro `n` e uma lista `l` e retorna todos os elementos de `l` que são menores que `n`, e `maiores` recebe um inteiro `n` e uma lista `l` e retorna todos os elementos de `l` que são maiores que `n`. Como modificar o `quickSort` para que ele devolva a lista ordenada sem elementos repetidos? Como modificar o `quickSort` para que ele devolva a lista ordenada de forma decrescente?