Aula Teórico-prática 6

Programação Funcional

LEI 1º ano

1. Apresente definições das funções sobre listas, já predefinidas no Prelude,

   ```
   takeWhile, dropWhile :: (a->Bool) -> [a] -> [a]
   ```

2. Defina uma função `break :: (a-> Bool) -> [a] -> ([a],[a])`, que calcula simultaneamente estes dois resultados. Note que apesar de poder ser definida à custa das outras duas, usando a definição

   ```
   break p l = (takeWhile p l, dropWhile p l)
   ```

   essa definição não é muito *eficiente*. `break` é também ums função predefinida no Prelude.

3. Revisite a ficha de exercícios da aula prática 4, e analise as funções que definiu na resolução dessa ficha.

   Relembre as funções de ordem superior (map, filter, foldr,...) que estudou e, sempre que achar apropriado, utilize-as para definir versões alternativas para funções que aí são pedidas.

4. Considere a função seguinte

   ```
   indicativo  :: String -> [String] -> [String]
   indicativo ind telefs = filter (concorda ind) telefs
     where concorda :: String -> String -> Bool
           concorda [] _ = True
           concorda (x:xs) (y:ys) = (x==y) && (concorda xs ys)
           concorda (x:xs) [] = False
   ```

   que recebe uma lista de algarismos com um indicativo, uma lista de listas de algarismos representando números de telefone, e seleciona os números que começam com o indicativo dado. Por exemplo:

   ```
   indicativo "253" ["253116787","213448023","253119905"]
   ```

   devolve `["253116787","253119905"]`.

   Redefina esta função com recursividade explícita, isto é, evitando a utilização de `filter`.