# Aula Teórico-prática e Prática 6

Programação Funcional

LCC 1º ano

1. A função `span :: (a->Bool) -> [a] -> ([a],[a])` pode ser definida como

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

   Apresente uma definição alternativa de `span` que percorra a lista uma única vez.

2. A função `break :: (a-Bool) -> [a] -> ([a],[a])` pode ser definida como

   ```
   break p l = span (not.p) l
   ```

   Apresente uma definição alternativa de `break` que não use a função `span`

3. Use as funções `break` e `dropWhile` para definir a função `words :: String -> String` que parte um texto nas suas palavras.

4. Relembre o problema da aula anterior para representar linhas poligonais. Nessa altura definimos uma linha como uma sequência de pontos. Sobre essa representação definimos funções de translação e de duplicação de linhas poligonais.

   Uma representação alternativa consiste em representar as linhas com um ponto de origem e uma sequência de vectores.

   ```
   type Poly = (Ponto,[Vector])
   Type Vector = (Float,Float)
   ```

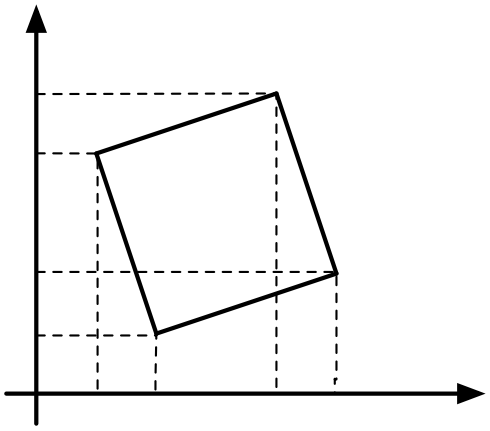
   Assim, o quadrado

   ```
   [(2,1), (5,2), (4,5), (1,4), (2,1)]
   ```

   poderia ser representado pelo ponto inicial `(2,1)` e pela seguinte lista de vectores

   ```
   [(3,1),(-1,3), (-3,-1), (1,-3)]
   ```

   Note que nesta representação, as funções de translação e duplicação de uma linha poligonal são muito mais simples.

   (a) Use a função `zipWith` para definir a função de conversão de elementos do tipo `Poligonal` em `Poly`.

   (b) Defina ainda a função que faz a conversão inversa.

   (c) Use estas funções de conversão para redefinir as funções de translação e duplicação de poligonais.