Exame de Programação Funcional – 1° Ano, MIEI / LCC / MIEF
30 de Janeiro de 2017 (Duração: 2 horas)

1. Apresente uma definição recursiva das seguintes funções (pré-definidas) sobre listas:
   (a) `unlines :: [String] -> String` que junta todas as strings da lista numa só, separando-as pelo caracter `'\n'`. Por exemplo, `unlines ["Prog", "Func" == "Prog\nFunc"`.
   (b) `(\\) :: (Eq a) => [a] -> [a] -> [a]` que retorna a lista resultante de remover (as primeiras ocorrências) dos elementos da segunda lista da primeira. Por exemplo,
   `(\\) [1,2,3,4,5,1] [1,5] == [2,3,4,1]` e
   `(\\) [1,2,2,3,2,1,4,1] [2,1,2] == [3,2,1,4,1]`.

2. Considere o seguinte tipo de dados para representar uma sequência em que os elementos podem ser acrescentados à esquerda (`Inicio`) ou à direita (`Fim`) da sequência.

```
data Seq a = Nil | Inicio a (Seq a) | Fim (Seq a) a
```

   (a) Defina a função `primeiro :: Seq a -> a` que recebe uma sequência não vazia e devolve o primeiro elemento.
   (b) Defina a função `semUltimo :: Seq a -> Seq a` que recebe uma sequência não vazia e devolve a sequência sem o seu último elemento.

3. Considere o seguinte tipo para representar árvores binárias.

```
data BTree a = Empty | Node a (BTree a) (BTree a)
```

   (a) Defina uma função `prune :: Int -> BTree a -> BTree a`, que remove de uma árvore todos os elementos a partir de uma determinada profundidade.
   (b) Defina uma função `semMinimo :: (Ord a) => BTree a -> BTree a` que remove o menor elemento de uma **árvore binária de procura não vazia**.

4. O problema das N rainhas consiste em colocar N rainhas num tabuleiro de xadrez com N linhas e N colunas, de tal forma que nenhuma rainha está ameaçada por outra. Note que uma rainha ameaça todas as posições que estão na mesma linha, na mesma coluna ou nas mesmas diagonais.
   Uma forma de representar estas soluções é usando listas de strings. O exemplo representa uma solução para este problema quando N é 4.

```
type Tabuleiro = [String]

exemplo :: Tabuleiro
exemplo = ["..R.",
           "R...",
           "...R",
           ".R.."]
```

   (a) Defina a função `posicoes :: Tabuleiro -> [(Int,Int)]` que determina as posições (coluna e linha) onde se encontram as rainhas num tabuleiro, de tal forma que `posicoes exemplo == [(2,0),(0,1),(3,2),(1,3)]`.
   (b) Usando a função anterior, defina a função `valido :: Tabuleiro -> Bool` que testa se num tabuleiro nenhuma rainha ataca outra. No caso do tabuleiro exemplo a resposta deve ser True. Note que pode testar se duas rainhas estão na mesma diagonal vendo se a soma ou a diferença entre a linha e a coluna em que estão colocadas são iguais.
   (c) Utilizando funções de ordem superior, defina a função `bemFormado :: Int -> Tabuleiro -> Bool` que dado um tamanho n e um tabuleiro t testa se este é bem formado, isto é, tem n linhas, n colunas, n rainhas, e os restantes caracteres do tabuleiro são o '.' .