Teste Final de Programação Funcional – 1° Ano, MIEI / LCC / MIEF
5 de Janeiro de 2019 (Duração: 2 horas)

1. Apresente uma definição recursiva das seguintes funções (pré-definidas) sobre listas:

   (a) `elemIndices :: Eq a => a -> [a] -> [Int]` que calcula a lista de posições em que um elemento ocorre numa lista. Por exemplo, `elemIndices 3 [1,2,3,4,3,2,3,4,5]` corresponde a `[2,4,6]`.

   (b) `isSubsequenceOf :: Eq a => [a] -> [a] -> Bool` que testa se os elementos de uma lista ocorrem noutra pela mesma ordem relativa. Por exemplo, `isSubsequenceOf [20,40] [10,20,30,40]` corresponde a `True` enquanto que `isSubsequenceOf [40,20] [10,20,30,40]` corresponde a `False`.

2. Considere o seguinte tipo para representar árvores binárias.

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

   (a) Defina a função `lookupAP :: Ord a => a -> BTree (a,b) -> Maybe b` que generaliza função `lookup` para árvores binárias de procura.

   (b) Defina a função `zipWithBT :: (a -> b -> c) -> BTree a -> BTree b -> BTree c` que generaliza a função `zipWith` para árvores binárias.

3. Defina a função `digitAlpha :: String -> (String,String)`, que dada uma string, devolve um par de strings: uma apenas com os números presentes nessa string, e a outra apenas com as letras presentes na string. Implemente a função de modo a fazer uma única travessia da string. Sugestão: pode usar as funções `isDigit, isAlpha :: Char -> Bool`.

4. Considere o seguinte tipo de dados para representar uma sequência em que os elementos podem ser acrecentados à esquerda (`Cons`) ou por concatenação de duas sequências (`App`).

   ```
   data Seq a = Nil | Cons a (Seq a) | App (Seq a) (Seq a)
   ```

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

   (b) Defina a função `dropSeq :: Int -> Seq a -> Seq a`, tal que `dropSeq n s` elimina os `n` primeiros elementos da sequência `s`. A função deve manter a estrutura da sequência.
   Por exemplo: `dropSeq 2 (App (App (Cons 7 (Cons 5 Nil)) (Cons 3 Nil)) (Cons 1 Nil)) == App (Cons 3 Nil) (Cons 1 Nil)`

   (c) Declare `(Seq a)` como instância da classe `Show` de forma a obter o seguinte comportamento no interpretador:

   ```
   > App (Cons 1 Nil) (App (Cons 7 (Cons 5 Nil)) (Cons 3 Nil))
   <<1,7,5,3>>
   ```

5. Considere a sequinte definição para representar matrizes: `type Mat a = [[a]]`
   Por exemplo, a matriz $\begin{bmatrix} 6 & 7 & 2 \\ 1 & 5 & 9 \\ 8 & 3 & 4 \end{bmatrix}$ seria representada por `[[6,7,2], [1,5,9], [8,3,4]]`

   (a) Defina a função `getElem :: Mat a -> IO a`, que selecciona aleatoriamente um elemento da matriz. Sugestão: use a função `randomRIO :: Random a => (a,a) -> IO a`.

   (b) Um quadrado mágico é uma matriz quadrada de inteiros em que a soma de qualquer linha, coluna e das duas diagonais é uma constante. O exemplo acima é um quadrado mágico com constante 15.
   Defina a função `magic :: Mat Int -> Bool` que verifica se uma matriz quadrada é um quadradro mágico. Será valorizada a utilização de funções de ordem superior.