**Exame de Programação Funcional** – 1º Ano, MIEI / LCC / MIEF
25 de Janeiro de 2020 (Duração: 2 horas)

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

   (a) `inits:: [a] -> [[a]]` que calcula a lista dos prefixos de uma lista. Por exemplo, `inits [11,21,13]` corresponde a `[[],[11],[11,21],[11,21,13]]`.

   (b) `isPrefixOf:: Eq a => [a] -> [a] -> Bool` que testa se uma lista é prefixo de outra. Por exemplo, `isPrefixOf [10,20] [10,20,30]` corresponde a `True` enquanto que `isPrefixOf [10,30] [10,20,30]` corresponde a `False`.

2. Considere o seguinte tipo para representar árvores binárias.

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

   (a) Defina a função `folhas :: BTree a -> Int`, que calcula o número de folhas (i.e., nodos sem descendentes) da árvore.

   (b) Defina a função `path :: [Bool] -> BTree a -> [a]`, que dado um caminho (`False` corresponde a *esquerda* e `True` a *direita*) e uma árvore, dá a lista com a informação dos nodos por onde esse caminho passa.

3. Uma representação possível de polimómios é pela sequência dos coeficientes - têm que se armazenar também os coeficientes nulos pois será a posição do coeficiente na lista que dará o grau do monómio.

```
type Polinomio = [Coeficiente]
type Coeficiente = Float
```

   A representação do polinómio $2\,x^5 - 5\,x^3$ será então `[0,0,0,-5,0,2]`, que corresponde ao polinómio $0\,x^0 + 0\,x^1 + 0\,x^2 - 5\,x^3 + 0\,x^4 + 2\,x^5$. Nas questões que se seguem, use sempre que possível, funções de ordem superior.

   (a) Defina a operação `valor :: Polinomio -> Float -> Float` que calcula o valor do polinómio para um dado $x$.

   (b) Defina a operação `deriv :: Polinomio -> Polinomio` que calcula a derivada de um polinómio.

   (c) Defina a operação `soma :: Polinomio -> Polinomio -> Polinomio` de adição de polinómios.

4. Considere a seguinte definição para representar matrizes: `type Mat a = [[a]]`.

   `ex = [[1,4,3,2,5], [6,7,8,9,0], [3,5,4,9,1]]` representa a matriz abaixo desenhada.

   (a) Defina a função `quebraLinha :: [Int] -> [a] -> [[a]]` que recebe uma lista de inteiros `s` e uma linha `l`, e produz a lista de segmentos contíguos de `l` de comprimento indicado em `s`. Por exemplo, `quebraLinha [2,3] [1,4,3,2,5] == [[1,4],[3,2,5]]`.

| | | | | |
|---|---|---|---|---|
| 1 | 4 | 3 | 2 | 5 |
| 6 | 7 | 8 | 9 | 0 |
| 3 | 5 | 4 | 9 | 1 |

   (b) Defina a função `fragmenta :: [Int] -> [Int] -> Mat a -> [Mat a]` que recebe duas lista de inteiros (com a partição das linhas e das colunas) e uma matriz, e produz a lista de (sub)-matrizes de acordo com essa partição. Por exemplo,
   `fragmenta [2,1] [2,3] ex == [ [[1,4],[6,7]], [[3,2,5],[8,9,0]], [[3,5]], [[4,9,1]] ].`

   (c) Defina a função `geraMat :: (Int,Int) -> (Int,Int) -> IO (Mat Int)` tal que `geraMat (x,y) (a,b)` gera aleatoriamente uma matriz com `x` linhas e `y` colunas, cujos valores estão compreendidos entre `a` e `b`. Sugestão: use a função `randomRIO :: Random a => (a,a) -> IO a`.