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

1. Considere o tipo `MSet a` para representar multi-conjuntos de tipo `a`

   ```
   type MSet a = [(a,Int)]
   ```

   Considere ainda que nestas listas não há pares cuja primeira componente coincida, nem cuja segunda componente seja menor ou igual a zero. Para além disso, os multi-conjuntos estão organizados por ordem decrescente da muiltplicidade. O multi-conjunto {`'b','a','c','b','b','a','b'`} é representado pela lista `[('b',4),('a',2),('c',1)]`, por exemplo.

   (a) Defina a função `cardMSet :: MSet a -> Int` que calcula a cardinalidade de um multi-conjunto. Por exemplo, `cardMSet [('b',4),('a',2),('c',1)]` devolve 7.

   (b) Defina a função `moda :: MSet a -> [a]` que devolve a lista dos elementos com maior número de ocorrências.

   (c) Defina a função `converteMSet :: MSet a -> [a]` que converte um multi-conjunto numa lista. Por exemplo, `converteMSet [('b',4),('a',2),('c',1)]` devolve ``''bbbbaac''``.

   (d) Defina a função `addNcopies :: Eq a => MSet a -> a -> Int -> MSet a` que faz a inserção de um dado número de ocorrências de um elemento no multi-conjunto, mantendo a ordenação por ordem decrescente da multiplicidade. Não use uma função de ordenação.

2. Considere o seguinte tipo de dados para representar subconjuntos de números reais (`Doubles`).

   ```
   data SReais = AA Double Double | FF Double Double
               | AF Double Double | FA Double Double
               | Uniao SReais SReais
   ```

   `(AA x y)` representa o intervalo aberto $]x, y[$, `(FF x y)` representa o intervalo fechado $[x, y]$, `(AF x y)` representa $]x, y]$, `(FA x y)` representa $[x, y[$ e `(Uniao a b)` a união de conjuntos.

   (a) Defina a `SReais` como instância da classe `Show`, de forma a que, por exemplo, a apresentação do termo `Uniao (Uniao (AA 4.2 5.5) (AF 3.1 7.0)) (FF (-12.3) 30.0)` seja

   ```
   ((]4.2,5.5[ U ]3.1,7.0]) U [-12.3,30.0])
   ```

   (b) Defina a função `pertence :: Double-> SReais -> Bool` que testa se um elemento pertence a um conjunto.

   (c) Defina a função `tira :: Double -> SReais -> SReais` que retira um elemento de um conjunto.

3. Considere o seguinte tipo para representar árvores irregulares (*rose trees*).

   ```
   data RTree a = R a [RTree a]
   ```

   (a) Defina a função `percorre :: [Int] -> RTree a -> Maybe [a]` que recebe um caminho e uma árvore e dá a lista de valores por onde esse caminho passa. Se o caminho não for válido a função deve retornar `Nothing`. O caminho é representado por uma lista de inteiros (1 indica seguir pela primeira sub-árvore, 2 pela segunda, etc)

   (b) Defina a função `procura :: Eq a => a -> RTree a -> Maybe [Int]` que procura um elemento numa árvore e, em caso de sucesso, calcula o caminho correspondente.