Programação Funcional
1° Ano – LEI/LCC

17 de Janeiro de 2011 – Duração: 2 horas

Teste

Parte I

Esta parte do teste representa 12 valores da cotação total. Cada alínea está cotada em 2 valores. **A não obtenção de uma classificação mínima de 8 valores nesta parte implica a reprovação no teste.**

1. Um multiconjunto é um conjunto em que a multiplicidade é relevante.

   São por isso diferentes os multiconjuntos: $\{1,2,3,4,1,2,1\}$ e $\{1,2,3,4\}$. Considere que se usa o seguinte tipo para representar multi-conjuntos:

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

   Neste tipo, o multiconjunto `{'a','c','a','b','c','a'}` é representado por `[('a',3), (´'b',1), ('c', 2)]`. Considere ainda que estas listas estão ordenadas (pela primeira componente) e que não há pares cuja primeira componente coincida.

   (a) A função `melem :: Ord a => a -> MSet a -> Bool` testa se um elemento pertence a um multiconjunto. Uma possível definição dessa função é:

   ```
   melem x m = not (null (filter (aux x) m))
   ```

   Defina a função `aux` usada nesta definição.

   (b) Apresente uma definição alternativa dafunção `melem` usando recursividade explícita.

   (c) Defina a função de ordem superior `mfilter :: (a -> Bool) -> MSet a -> MSet a` que filtra num multi-conjunto os elementos que satisfazem um dado predicado.

   (d) Defina uma função `media :: MSet Float -> Float` que calcula a média dos elementos de um multiconjunto (não vazio). Note que o número de vezes que um elemento aparece num multiconjunto é relevante para o valor da média,

2. Considere o seguinte tipo de dados:

   ```
   data From a = Last a | Next (From a)
   ```

   (a) Este tipo é isomorfo a um tuplo com um `a` (armazenado no construtor `Last`) e um número natural (o número de constructores `Next`). Defina uma função `topair :: From a -> (a,Int)` que converte um valor do tipo `From a` no tuplo respectivo. Por exemplo, `topair (Next (Next (Next (From 'a')))) == ('a',3)`.

   (b) Defina uma instância da classe `Show` para o tipo `From a`, de tal forma que `topair (Next (Next (Next (From 'a')))) == "+++'a'"` (ou seja, o elemento contido num `From a` seja precedido por tantos `+` quanto o número de constructores `Next`).

Parte II

1. Considere o seguinte tipo:

   ```
   data List a b = Nil | ConsA a (List a b) | ConsB b (List a b)
   ```

   (a) Escreva uma função `unzipAB :: List a b -> ([a],[b])` que produz as listas com todos os elementos dos tipos `a` e `b` contidos numa `List a b`

   (b) Escreva uma função `sortA :: (Ord a) => List a b -> List a b` que ordena apenas os elementos de tipo `a` da lista, mantendo os elementos de tipo `b` inalterados.

2. O conjunto dos números racionais é isomorfo ao conjunto dos números naturais. Este facto implica, por exemplo, que é possível codificar qualquer par de números naturais num único número natural. O método de Cantor é um dos possíveis métodos para efectuar essa codificação: dado par de naturais (i, j) o número que lhe corresponde é dado pela seguinte matriz.

$\frac{i}{j}$	0	1	2	3	$\cdots$
0	0	1	3	6	
1	2	4	7	11	
2	5	8	12	17	
3	9	13	18	24	
$\vdots$					

   Defina as funções

   ```
   pair :: (Integer,Integer) -> Integer
   unpair :: Integer -> (Integer,Integer)
   ```

   que implementam este mecanismo de codificação de pares. Por exemplo, de acordo com a tabela acima, `pair (1,3) == 13` e `unpair 7 == (2,1)`.