Exame

Programação Funcional – 1° Ano, LEI / LCC 15 de Fevereiro de 2013

Duração: 2 horas

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. Defina a função sufixos :: [a] -> [[a]] que, dada uma lista dá como resultado os sufixos dessa lista. Por exemplo, sufixos [1,2,3] = [[1,2,3],[2,3],[3],[]].
- Defina a função dropWhile :: (a->Bool) -> [a] -> [a] que vai retirando os elementos iniciais de uma lista enquanto um dado predicado é satisfeito. Por exemplo: dropWhile (< 8) [6,1,5,4,9,3,8,2,0,9,1] = [9,3,8,2,0,9,1].
- 3. Assuma que as notas dos alunos estão armazenadas numa árvore binária, declarada da seguinte forma:

```
data Alunos = Vazia | Nodo (Numero, Nome, Nota) Alunos Alunos
type Numero = Int
type Nome = String
type Nota = Int
```

(a) Considere a definição da função aprovados que calcula a percentagem de alunos aprovados.

Defina a função conta e indique o seu tipo.

- (b) Assumindo que Alunos é uma árvore binária de procura (ordenada por número de aluno), defina a função nota :: Numero -> Alunos -> Maybe Nota, que permite saber a nota de um dado aluno, caso exista.
- 4. Para armazenar conjuntos de números inteiros, optou-se pelo uso de sequências de intervalos.

```
type ConjInt = [Intervalo]
type Intervalo = (Int,Int)
```

Assim, por exemplo, o conjunto $\{1, 2, 3, 4, 7, 8, 19, 21, 22, 23\}$ poderia ser representado por [(1,4),(7,8),(19,19),(21,23)].

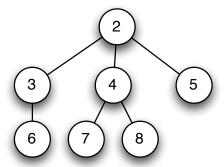
- (a) Defina a função pertence :: Int -> ConjInt -> Bool que testa se um dado inteiro pertence a um conjunto.
- (b) Defina uma função quantos :: ConjInt -> Int que, dado um conjunto, dê como resultado o número de elementos desse conjunto. Por exemplo, quantos [(1,4),(7,8),(19,19),(21,23)] = 10.

Parte II

- 1. Recorde a estrutura de dados ConjInt, apresentada na alínea 4 da Parte I, para representar conjuntos de números inteiros.
 - (a) Defina uma função elems :: ConjInt -> [Int] que, dado um conjunto, dâ como resultado a lista dos elementos desse conjunto. Por exemplo, elems [(1,4),(7,8),(19,19),(21,23)] = [1,2,3,4,7,8,19,21,22,23].
 - (b) Defina uma função geraconj :: [Int] -> ConjInt que recebe uma lista de inteiros, ordenada pordem crescente e sem repetições, e gera um conjunto. Por exemplo, geraconj [1,2,3,4,7,8,19,21,22,23] = [(1,4),(7,8),(19,19),(21,23)].
- 2. Considere a seguinte definição para representar árvores irregulares

data ArvIrr a = No a [ArvIrr a]

A expressão No 2 [No 3 [No 6 []], No 4 [No 7 [], No 8 []], No 5 []] representa a árvore



- (a) Defina a função maximo :: Ord a => ArvIrr a -> a que calcula a maior elemento de uma destas árvores.
- (b) A função elems, definida abaixo, lista os elementos de uma árvore, bem como o nível a que aparecem (segundo uma travessia pre-order).

Por exemplo, para a árvore apresentada, a função retorna a lista

$$[(2,1),(3,2),(6,3),(4,2),(7,3),(8,3),(5,2)]$$

elems a = elemsAux 1 a

where elemsAux n (No x 1) = (x,n):(concat (map (elemsAux (n+1)) 1))

Defina a função unElems :: [(a,Int)] -> ArvIrr a inversa da anterior, no sentido em que, para toda a árvore a, unElems (elems a) = a.