EXAME - 2ª Chamada
1·Fevereiro·2005
Duração: 2:00 horas

Paradigmas de Programação I
LESI

Nº: _____ **NOME:** _____

I

Considere as declarações da classe `FigFechada` e da função `fun` a seguir apresentadas

```
class FigFechada a where
  area :: a -> Float
  perimetro :: a -> Float

fun figs = filter (\fig -> (area fig) > 100) figs
```

1. Indique, justificado, qual é o tipo inferido pelo interpretador Haskell para a função `fun`.

2. No plano cartesiano, um rectângulo com os lados paralelos aos eixos podem ser univocamente determinado pelas coordenadas do vértice inferior esquerdo e pelos comprimentos dos lados, ou por uma diagonal dada por dois pontos. Assim, para representar esta figura geométrica, definiu-se em Haskell o seguinte tipo de dados:

   ```
   type Ponto = (Float,Float)
   type Lado = Float
   data Rectangulo = PP Ponto Ponto
                   | PLL Ponto Lado Lado
   ```

   Declare `Rectangulo` como instância da classe `FigFechada`.

3. Defina a função `somaAreas :: [Rectangulo] -> Float` que calcula o somatório das áreas de uma lista de rectângulos. (De preferência, utilize funções de ordem superior.)

Nº: _____ **NOME:** _____

II

Relembre a definição de árvores binárias.

```
data BTree a = Vazia
             | Nodo a (BTree a) (BTree a)
```

1. Uma árvore binária diz-se de procura se uma travessia inorder da árvore resultar numa lista ordenada. Usando a função que a seguir se apresenta, defina uma função que teste se uma árvore é de procura.

   ```
   inorder Vazia = []
   inorder (Nodo r e d) = (inorder e) ++ (r:(inorder d))
   ```

2. Numa árvore de procura (não vazia), o menor elemento pode ser calculado usando a seguinte função:

   ```
   menor (Nodo r Vazia _) = r
   menor (Nodo _ e _) = menor e
   ```

   Defina agora uma função que, dada uma árvore não vazia, retorne um par cuja primeira componente é o menor elemento da árvore e a segunda é a árvore sem esse elemento.

3. Para remover um elemento de uma árvore binária de procura podemos usar a seguinte estratégia:

   - Se o elemento a remover é a raiz de uma árvore cuja sub-árvore direita é vazia então o resultado é a sub-árvore esquerda.
   - No outro caso, basta remover o menor elemento da sub-árvore direita e colocá-lo na raiz em substituição do elemento a remover.

   Usando a função da alínea anterior, defina uma função que remova um elemento de uma árvore binária de procura.

Nº: _____ NOME: _____

III

Relembre o projecto prático e considere que se fizeram as seguintes simplificações na representação dos dados.

```
type Concorrentes = [(Int, String)]          -- número e nome
type Prova = [(Int, Int)]                    -- num concorente e posição
```

Considere ainda que existe definida uma função `sort` que ordena uma dada lista por ordem crescente.

1. Defina uma função

   ```
   junta :: Prova -> Concorrentes -> [(Int,String,Int)]
   ```

   que junte a informação das duas tabelas de informação.

2. Supondo que não incluía no código a informação do tipo da função que definiu, qual seria o tipo inferido pelo interpretador ?

3. Considere agora que as duas tabelas que correspondem aos tipos acima se encontram armazenadas em ficheiro, e assuma que já existem as funções

   ```
   geraConcorrentes :: String -> Concorrentes
   geraProva :: String -> Prova
   ```

   para converter a informação que vem de ficheiro.

   (a) Defina uma função que, dados os nomes dos dois ficheiros, escreve no ecran a lista dos nomes dos vários concorrentes, ordenados por posição na prova.

   (b) Defina agora, e usando a função anterior, um programa (i.e., do tipo `IO ()`) que pergunte ao utilizador quais os nomes dos ficheiros onde se encontram os dados e depois invoque a função da alínea anterior para produzir o resultado aí apontado.