EXAME - Recurso
18·Fevereiro·2005
Duração: 2:00 horas

Paradigmas de Programação I
LESI

Nº: _____ **NOME:** _____

I

Recorde o que aprendeu sobre tipos de dados e classes em Haskell.

1. Defina um tipo de dados que represente a indicação horária num relógio analógico i.e. com uma representação do tipo `05:43AM` ou `04:35PM` .

2. Declare o tipo que definiu como instância da classe `Eq`, fazendo coincidir os valores `00:00AM` e `12:00PM`, e os valores `00:00PM` e `12:00AM`.

3. Recorde a classe `Enum`:

```
class Enum a where
    pred :: a -> a                          {- has default method -}
    succ :: a -> a                          {- has default method -}
    toEnum :: Int -> a
    fromEnum :: a -> Int
    enumFrom :: a -> [a]                    {- has default method -}
    enumFromThen :: a -> a -> [a]           {- has default method -}
    enumFromTo :: a -> a -> [a]             {- has default method -}
    enumFromThenTo :: a -> a -> a -> [a] {- has default method -}
```

   Declare o tipo de dados que definiu como instância da classe `Enum`, de forma a que `succ` avance o relógio 1 minuto e `pred` recue o relógio 1 minuto.

Notas:

(i) o sucessor de `11:59PM` deverá ser `00:00AM` .

(ii) a definição mínima de uma instância da classe `Enum` exige apenas que os métodos `toEnum` e `fromEnum` estejam definidos.

(iii) as funções `pred` e `succ` estão definidas como:

```
pred x = (toEnum (fromEnum x) - 1)
succ x = (toEnum (fromEnum x) + 1)
```

Nº: _____ **NOME:** _____

II

Uma forma de representar polinómios de uma variável é usar listas de pares *(coeficiente, expoente)*

```
type Pol = [(Float,Integer)]
```

Note que o polinómio pode não estar simplificado. Por exemplo,

```
[(2.4,3), (3.0,4), (5.2,3), (4.1,5)]  :: Pol
```

representa o polinómio $2.4\,x^3 + 3.0\,x^4 + 5.2\,x^3 + 4.1\,x^5$.

1. Defina uma função

   ```
   simp :: Pol -> Pol
   ```

   que faça a simplificação de um polinómio, isto é, que dado um polinómio construa um polinómio equivalente ao primeiro em que não podem aparecer varios monómios com o mesmo grau.

2. Defina uma função

   ```
   ordena :: Pol -> Pol
   ```

   que dado um polinómio construa um polinómio equivalente ao primeiro em que os monómios surgem na lista por ordem crescente de grau.

3. Defina uma função

   ```
   equiv :: Pol -> Pol -> Bool
   ```

   que teste se dois polinómios são equivalentes.

Nº: _____ NOME: _____

III

Para implementar tabelas como árvores binárias de procura com pares *(chave,valor)* nos nódos da árvore, definiu-se o seguinte tipo de dados:

```
data Tabela a b = Vazia
                | Nodo (a,b) (Tabela a b) (Tabela a b)
```

Definiram-se também as seguintes funções:

```
nova = Vazia

remove _ Vazia = Vazia
remove x (Nodo (c,_) e Vazia) | x == c = e
remove x (Nodo (c,_) Vazia d) | x == c = d
remove x (Nodo (c,v) e d)
             | x < c  = Nodo (c,v) (remove x e) d
             | x > c  = Nodo (c,v) e (remove x d)
             | x == c = let  (y,z) = minTab d
                        in Nodo (y,z) e (remove y d)

minTab :: Tabela a b  -> (a,b)
minTab (Nodo (c,v) Vazia _) = (c,v)
minTab (Nodo _ e _)         = minTab e
```

1. Indique, justificado, qual é o tipo inferido pelo interpretador Haskell para as funções `nova` e `remove`.

2. Defina a função `procura :: (Ord a) => a -> Tabela a b -> Maybe b` que dada uma chave e uma tabela, devolve o valor associado a essa chave na tabela (caso a chave exista).

3. Defina a função `actualiza :: (Ord a) => (a,b) -> Tabela a b -> Tabela a b` que insere uma associação *(chave,valor)* numa dada tabela. Se a chave já existir na tabela, a nova associação sobrepõe-se à que está na árvore.

4. Indique o que entende por *Tipo Abstracto de Dados*, e o que deve fazer para construir o tipo abstracto `Tabela` que tenha como *interface* as funções: `nova`, `actualiza`, `procura` e `remove`.