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

Programação Funcional
Paradigmas de Programação I
LMCC

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,

```
p1 = [(2.4,3), (3.0,4), (5.2,3), (4.1,5)]
```

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

1. Defina a função

   ```
   apresenta :: Pol -> String
   ```

   de forma a que

   ```
   Main> apresenta p1
   ‘‘2.4 x^3 + 3.0 x^4 + 5.2 x^3 + 4.1 x^5’’
   ```

2. 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.

3. 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.

4. Defina uma função

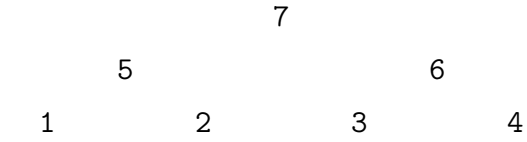
   ```
   equiv :: Pol -> Pol -> Bool
   ```

   que teste se dois polinómios são equivalentes.

Nº: _____ **NOME:** _____

III

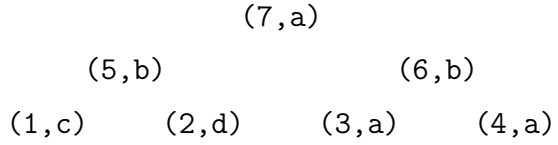
Pretende-se construir uma função de marcação de árvores em que as marcas são definidas sequencialmente nos elementos de uma lista; o exemplo seguinte ilustra o algorimo de marcação pretendido.



$+$

$[\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}, \texttt{b}, \texttt{a}, \texttt{a}]$

$=$



Sugestão: *Recorde as funções de marcação de árvores pela ordem "preorder" em que se associa a cada nodo da árvore uma ordem de visita sob a forma de um inteiro.*

1. Construa uma função que tome como argumentos uma árvore com nodos de um tipo arbitrário e uma lista com elementos também de tipo arbitrário e construa a árvore marcada conforma atrás indicado. Suponha que o número de elementos da lista é suficiente para marcar toda a árvore.

2. Se se considerar listas de tamanho arbitrário, a função deve falhar se a lista não tiver elementos suficientes para marcar toda a árvore. Modifique a função definida na alínea anterior de forma a contemplar esta possibilidade.

3. Suponha que já construi a árvore de inteiros marcada por uma sequência de valores booleanos; defina uma função que, usando o iterador `iter`, calcule a soma dos elementos da árvore que estão associados à marca `True`.