Paradigmas da Programação I / Programação Funcional

ESI / MCC

Ano Lectivo de 2005/2006 (Recurso)

Questão 1 Considere o seguinte tipo de dados que descreve a informação de um extracto bancário. Cada valor deste tipo indica o saldo inicial (valor `Int`), e uma lista (não ordenada) de movimentos. Cada movimento é representado por um triplo que indica a data da operação (valor `Data`), a sua descrição (valor `String`) e a quantia movimentada (valor `Movimento` em que os inteiros são sempre números positivos).

```
data Data = DiaMesAno Int Int Int
data Movimento = Credito Int | Debito Int
data Extracto = Ext Int [(Data, String, Movimento)]
```

1. Construa a função `extValor :: Extracto -> Int -> [Movimento]` que produz uma lista de todos os movimentos superiores a um determinado valor. Por exemplo, se for passado o valor `10` no segundo parâmetro, a função deverá retornar todos os créditos e débitos com valor absoluto superior a `10`.

2. Defina a função `filtro :: Extracto -> [String] -> [(Data,Movimento)]` que retorna informação relativa apenas aos movimentos cuja descrição esteja incluída na lista fornecida no segundo parâmetro.

3. Utilize a função `foldr :: (a -> b -> b) -> b -> [a] -> b` para implementar a função `creDeb :: Extracto -> (Int,Int)` que retorna o total de créditos e de débitos num extracto no primeiro e segundo elementos de um par, respectivamente.

Questão 2 Pretende-se guardar informação sobre os aniversários das pessoas numa tabela que associa o nome de cada pessoa à sua data de nascimento. Para isso, declarou-se a seguinte estrutura de dados

```
type Dia = Int
type Mes = Int
type Ano = Int
type Nome = String

data Data = D Dia Mes Ano
   deriving Eq

type TabDN = [(Nome,Data)]
```

1. Defina a função `idade :: Data -> Nome -> TabDN -> Maybe Int`, que calcula a idade de uma pessoa numa dada data.

2. Declare o tipo `Data` como instância da classe `Ord`.

3. Defina a função `porIdade:: TabDN -> [Nome]` que dada a tabela de datas de nascimento, devolve a lista dos nomes ordenada por ordem crescente da idade das pessoas.

Questão 3 Definiu-se a seguinte estrutura de dados para armazenar a associação entre cada abreviatura e a sequência de caracteres que ela representa.

```
type Abreviaturas = [(String, String)]
```

Um exemplo de uma tabela destas pode ser

```
[("mto","muito"), ("/","mente"),  ("/a","menta"), ("c/","com")]
```

1. Defina a função `subst :: (String,String) -> String -> String` que substitui a abreviatura se ela estiver presente numa palavra. Por exemplo, o resultado de

   ```
   subst ("/a","menta") "imple/arem"
   ```

   deverá ser `"implementarem"`. Poderá utilizar a função `prefixo` abaixo definida como auxiliar. Assuma como simplificação que a palavra pode ter no máximo uma abreviatura.

   ```
   prefixo [] ys = True
   prefixo (_:_) [] = False
   prefixo (x:xs) (y:ys) = (x==y) && (prefixo xs ys)
   ```

2. Defina a função `trataTexto :: Abreviaturas -> [String] -> [String]` que faz a substituição das abreviaturas ao longo de um texto.

3. Use as funções acima para definir um programa `trataFich :: FilePath -> FilePath -> Abreviaturas -> IO ()` que recebe como argumentos os nomes de dois ficheiros (de entrada e de saída) e coloca no ficheiro de saída o texto do ficheiro de entrada depois de substituir todas as abreviaturas.