

# Paradigmas da Programação I / Programação Funcional

## ESI / MCC

Ano Lectivo de 2005/2006 (2a Chamada)

1

**Questão 1** Considere as seguintes declarações:

```
type Numero = String
type Nome = String
data Entrada = Ent Nome Numero
type LTelef = [Entrada]
data Telefonema = Tel Numero Numero      -- origem destino
type Registo = [Telefonema]
```

1. Defina a função nomeCliente:: LTelef -> Nome -> Maybe Entrada que determina a entrada correspondente a um dado nome (caso exista).
2. Defina a função numChamadas:: Nome -> Registo -> LTelef -> Int que determina o número de telefonemas efectuados por uma determinada pessoa.
3. Considere agora as declarações

```
type Email = String
data NovaEntrada = NEnt Nome Numero Email
type NovaLTelef = [NovaEntrada]
```

Defina a função actualizaLTelef:: LTelef -> [(Nome,Email)] -> NovaLTelef que cria uma NovaLTelef a partir de uma LTelef e de uma lista de emails. Para os nomes que não existam na lista de emails, vão ser gerados a partir dos números telefónicos (n. telefone 123 fica com o email e123@pp1.pt)

**Questão 2** 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 deriving Eq
data Movimento = Credito Int | Debito Int
data Extracto = Ext Int [(Data, String, Movimento)]
```

1. Declare o tipo Data como instância da classe Ord e construa a função extDatas :: Extracto -> Extracto que produz um extracto ordenado por datas.
2. Utilize a função foldr :: (a -> b -> b) -> b -> [a] -> b para implementar a função saldo :: Extracto -> Int que devolve o saldo final que resulta da execução de todos os movimentos no extracto sobre o saldo inicial.
3. Defina a função imprime :: Extracto -> IO () para visualizar a lista de movimentos. Para isso, comece por declarar os tipos Data e Movimento como instâncias da classe Show. Pretende-se que o extracto seja visualizado com o seguinte formato ("\\t" representa um carácter de tabulação e deve ser utilizado tal como indicado no exemplo):

```
Data\t\t\t\tDescricao\t\t\t\tDebito\t\t\t\tCredito
20-10-2001\t\t\t\tMultibanco\t\t\t\t-40\t\t\t\t0
27-10-2001\t\t\t\tSalario\t\t\t\t0\t\t\t\t1000
```

# Paradigmas da Programação I / Programação Funcional

## ESI / MCC

Ano Lectivo de 2005/2006 (2a Chamada)

---

2

**Questão 3** Uma árvore binária diz-se balanceada se

- é vazia ou
- os pesos (ou alturas) das sub-árvore esquerda e direita diferem no máximo de uma unidade, e ambas essas sub-árvore estão平衡adas.

Dada uma árvore binária, vamos construir uma outra árvore que contém, para cada nodo da árvore, o peso da árvore que aí se inicia.

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

```
pesos :: BTREE a -> BTREE (a, Int)
pesos Vazia = Vazia
pesos (Nodo x e d) = let pe = peso e
                      pd = peso d
                      p = 1 + (max pe pd)
                  in Nodo (x,p) (pesos e) (pesos d)
peso :: BTREE a -> Int
peso .....
```

1. Complete a definição anterior e defina uma função `eBalanceada :: BTREE a -> Bool` que determina se uma dada árvore está平衡ada.
2. A função `pesos` definida acima é muito pouco eficiente pois faz várias travessias da árvore. Uma outra alternativa para definir esta função consiste em usar uma função auxiliar que, ao calcular o peso da árvore, constrói ainda a árvore pretendida.

```
pesos a = snd (pesosAux a)
```

```
pesosAux Vazia = (0, Vazia)
pesosAux (Nodo r e d) = ...
```

3. Considere a seguinte definição:

```
filterBTREE :: (a -> Bool) -> (BTREE a) -> [a]
filterBTREE p a = filter p (inorder a)
  where inorder Vazia = []
        inorder (Nodo r e d) = (inorder e) ++ [r] ++ (inorder d)
```

Defina a função `filterBTREE` sem usar o passo intermédio de cálculo dos elementos da árvore. Dito de outra forma, complete a seguinte definição alternativa desta função:

```
filterBTREE p Vazia = ...
filterBTREE p (Nodo r e d) = ...
```