Métodos de Programação I

2.º Ano da LESI (5303O7) + LMCC (7003N5)
Ano Lectivo de 2005/06

Exame (1.ª chamada da época normal) de Janeiro 2007
09h30
Salas 2201 a 2208

NB*: Esta prova consta de **8** alíneas que valem, cada uma, 2.5 valores. Utilize folhas de resposta diferentes para cada grupo.*

PROVA SEM CONSULTA (2 horas)

GRUPO I

Questão 1 Dadas as funções

$$f = [id + i_1 \, , i_2 \cdot i_2] \tag{1}$$

e

$$f^\circ = [i_1 \cdot i_1 \, , i_2 + id] \tag{2}$$

identifique o isomorfismo que estas funções estabelecem (desenhando o diagrama respectivo) e derive a versão *pointwise* de f.

Questão 2 Prove que $f \cdot f^\circ = id$ se verifica, onde f e f° são as funções da questão anterior.

Questão 3 Considere o seguinte tipo de dados.

```
data Tree a = Leaf a | Child a (Tree a) | Node a (Tree a, Tree a)
```

Defina as funções `inTree`, `outTree`, `cataTree` e `anaTree` para este tipo de dados. Acompanhe as duas últimas definições com os diagramas respectivos.

GRUPO II

As duas questões que se seguem baseiam-se no tipo

```
data RList a = Nil | Cons (a , RList a)
```

da biblioteca `RList.hs`*, na qual, como sabe, se tem*

$$in = [\underline{Nil} \, , Cons] \tag{3}$$

$$map\, f = (\!|in \cdot (id + f \times id)|\!) \tag{4}$$

Questão 4 Demonstre que a função que se segue

```
unzip :: RList(a,b) -> (RList a, RList b)
unzip Nil = (Nil,Nil)
unzip (Cons((x,y),t)) = let (l,r) = unzip t
                        in (Cons(x,l), Cons(y,r))
```

pode ser definida em notação *pointfree* como $\langle map\ \pi_1, map\ \pi_2 \rangle$. **Sugestão:** use a lei de "banana-split".

Questão 5 Demonstre (por reflexão e fusão-cata) validade do facto

$$concat \cdot (map\ singl) = id \tag{5}$$

onde $singl$ é a função que constrói a lista singular com o seu argumento e

$$concat \quad = \quad (\!|[\underline{Nil}\ , cat]|\!) \tag{6}$$

onde cat é a função que concatena duas listas e satisfaz a propriedade

$$cat \cdot (singl \times id) = Cons \tag{7}$$

Questão 6 Suponha que o par de inteiros positivos (v, p) designa o número de bolas vermelhas (v) e pretas (p) que se encontram dentro de um saco, bolas essas que se vão tirando à sorte, sucessivamente, até o saco ficar vazio.

A função em Haskell que se segue

```
gera :: (Int,Int) -> Tree (Int,Int)
gera(0,0) = Leaf(0,0)
gera(v,0) = Child(v,0)(gera(v-1,0))
gera(0,p) = Child(0,p)(gera(0,p-1))
gera(v,p) = Node(v,p)(gera(v-1,p),gera(v,p-1))
```

representa sob a forma de uma árvore do tipo `Tree` da questão 3 todas as possíveis configurações do saco ao longo dessas experiências.

Codifique a função `gera` como um anamorfismo, desenhando o diagrama respectivo.

GRUPO III

Questão 7 Considere uma máquina de stack muito simples para calcular expressões aritméticas. Os comandos suportados por esta máquina são os seguintes:

```
push :: Int -> Comando ()
pop :: Comando Int
add :: Comando ()
mult ::  Comando ()
```

Como esperado, `push` insere um elemento no topo da stack, `pop` retira e devolve o elemento que está no topo da stack, e `add` e `mult` retiram dois elementos do topo da stack substituindo-os, respectivamente, pela sua soma e produto. Note que o comando `pop` nem sempre pode ser executado. Pretende-se implementar esta máquina usando o *monad* de estado, sendo o tipo dos comandos definido da seguinte forma:

```
type Stack = [Int]
type Comando a = StateT Stack Maybe a
```

Implemente os comandos acima referidos por forma a obter o seguinte comportamento.

```
> evalStateT (push 2 >> push 4 >> mult >> push 3 >> add >> pop) []
Just 9
> evalStateT (push 2 >> push 4 >> mult >> add >> pop) []
Nothing
```

No primeiro caso é calculada correctamente a expressão $3 + (4 * 2)$. No segundo caso tal não é possível pois quando é executado o comando `add` só existe um argumento na stack.

Questão 8 Recorde que um tipo paramétrico pode ser definido como uma mónada desde que se definam os operadores `join` (μ) e `return` (u), e se provem as propriedades

$$\begin{aligned} \mu \cdot \mu &= \mu \cdot \mathsf{F}\,\mu \\ \mu \cdot u &= \mu \cdot \mathsf{F}\,u = id \end{aligned}$$

Demonstre que $1 + A$ (cf. `Maybe A`) é uma mónada, isto é, apresente definições *pointfree* para μ (`join`) e u (`return`) deste tipo e prove as propriedades acima. **NB:** note que, para este tipo, $\mathsf{F}\,f = id + f$.
