Métodos de Programação I

2.º Ano de LESI (5303O7, Plano Antigo)
Ano Lectivo de 2007/08

Exame — 13 de Fevereiro de 2008
14h00
Sala 2105

NB*: Esta prova consta de **8** alíneas que valem, cada uma, 2.5 valores.*

PROVA SEM CONSULTA (2 horas)

GRUPO I

Questão 1 Dadas as funções

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

e

$$iso = [i_1 \cdot i_1 \, , i_2 + id] \tag{2}$$

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

Questão 2 Demonstre a propriedade natural da função iso da questão anterior.

$$(f + (g + h)) \cdot iso = iso \cdot ((f + g) + h) \tag{3}$$

Questão 3 Considere o tipo de dados indutivo das *listas não vazias*:

```
data NRList a = Sing a | Add  (a , NRList a)
```

Defina `inNRList`, `outNRList`, `cataNRList` e `anaNRList`. Acompanhe as suas definições com os diagramas respectivos.

GRUPO II

Questão 4 Relembre o tipo de listas polimórficas que está definido na biblioteca da disciplina.

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

Sobre este tipo é possível definir a função que inverte uma lista da seguinte forma

```
rev :: RList a -> RList a
rev Nil = Nil
rev (Cons h t) = snoc (h, rev t)
```

onde `snoc :: (a, RList a) -> Rlist a` é a função que insere um elemento no fim da lista. Demonstre que esta função pode ser definida como o seguinte catamorfismo sobre listas, justificando o melhor possível todos os passos da derivação (recorrendo, sempre que possível, às leis do cálculo estudado nesta disciplina).

```
rev = cataRList (either (const Nil) snoc)
```

Questão 5 Sobre o tipo `RList` é possível definir a função `map` que aplica uma função a todos os elementos de uma lista da seguinte forma:

```
mapRList f = cataRList (inRList . (id -|- f >< id))
```

Recorrendo, entre outras, à lei de fusão para listas, demonstre a seguinte propriedade de absorção para os catamorfismos deste tipo de dados, válida para quaisquer funções `f` e `g`.

```
cataRList g . mapRList f = cataRList (g . (id -|- f >< id))
```

Questão 6 Assumindo o tipo de dados indutivo das *listas não vazias* da questão 3, defina como um anamorfismo de `NRList` a função

```
tails :: [a] -> NRList [a]
```

que calcula os segmentos finais de uma lista. Por exemplo

```
> tails [1,2,3]
Add ([1,2,3],Add ([2,3],Add ([3],Sing [])))
```

GRUPO III

Questão 7 Considere o seguinte tipo de dados.

```
data BTree a = Empty | Node(a, (BTree a, BTree a))
```

Usando a móndada de estado, defina a função `decora :: BTree a -> BTree (a, Int)` que deverá etiquetar todos os nós da árvore argumento com um inteiro de tal forma que, numa travessia *inorder* da árvore decorada, os inteiros apareçam por ordem crescente.

Questão 8 Complete a seguinte demonstração de lei da associatividade da composição monádica:

$$f \bullet (g \bullet h) \quad = \quad (f \bullet g) \bullet h$$

$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \}$

$$f \bullet (\mu \cdot \mathsf{F}\, g \cdot h) \quad = \quad (\mu \cdot \mathsf{F}\, f \cdot g) \bullet h$$

$\equiv \quad \{ \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \}$

$\vdots$
