Métodos de Programação I

2.º Ano da LESI (5303O7) + LMCC (7003N5, Plano antigo)
Ano Lectivo de 2006/07

Exame (2.ª chamada da época normal) — 24 de Janeiro 2007
09h30
Salas 2202 a 2205

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 Dada a função

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

identifique o isomorfismo que esta função estabelece, desenhando o diagrama respectivo. Derive ainda a versão *pointwise* de iso.

Questão 2 Demonstre a propriedade natural da função iso da Questão 1:

$$(f \times (g + h)) \cdot iso \quad = \quad iso \cdot (f \times g + f \times h) \tag{2}$$

Questão 3 Considere o seguinte tipo de dados.

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

Defina as funções `inBTree`, `outBTree`, `cataBTree` e `anaBTree` para este tipo de dados. Acompanhe as duas últimas definições com os diagramas respectivos.

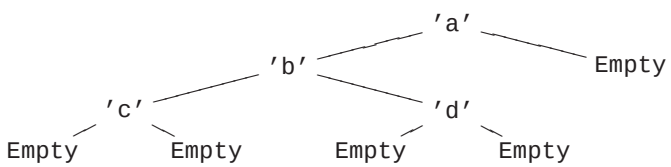
GRUPO II

Questão 4 Considere a função

```
rep :: BTree a -> [(Int, a)]
rep t = aux(t,1)

aux :: (BTree a, Int) -> [(Int, a)]
aux(Empty,i)          = []
aux(Node(a,(t,t')),i) = [(i, a)] ++ aux(t,2*i) ++ aux(t',2*i+1)
```

que representa árvores binárias sob a forma de 'array's modelados por listas de pares *(posição,elemento)*. Repare que, sempre que a i-ésima posição do *array* está ocupada com o habitante de um nó da árvore origem, então a sua sub-árvore esquerda é representada a partir da posição $2i$ e a sub-árvore direita a partir da posição $2i + 1$. Assim, a árvore



será representada pelo 'array'

```
[(1,'a'),(2,'b'),(4,'c'),(5,'d')]
```

Codifique a função `aux` como um hilomorfismo do tipo `BTree`, acompanhado a sua codificação de um diagrama ilustrativo.

Questão 5 Defina-se a função

```
caudas :: RList a -> RList (RList a)
caudas l = if nula l then Nil else Cons(l,caudas (cauda l))
```

no contexto da biblioteca `RList.hs`, onde se tem, como sabe

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

Demonstre que `caudas` pode ser definida como o anamorfismo de listas

$$[\!(\,(!+\langle id, cauda\rangle)\cdot nula?\,)\!] \tag{3}$$

onde $nula$ testa se uma lista é vazia e $cauda$ retorna a cauda de uma lista não vazia.
NB: relembre que $in = [\underline{Nil}\,, Cons]$.

Questão 6 Considere a função que, no contexto da bilioteca `RList.hs`, inverte uma lista

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

onde $singl\ a = Cons(a, Nil)$ e cat é a função que concatena duas `RLists` e que satisfaz a propriedade

$$cat \cdot (singl \times id) = Cons \tag{4}$$

É fácil mostrar que a transformada-PF de rev é o catamorfismo

$$rev = (\!|\,[\underline{Nil}\,, cat \cdot swap \cdot (singl \times id)]\,|\!) \tag{5}$$

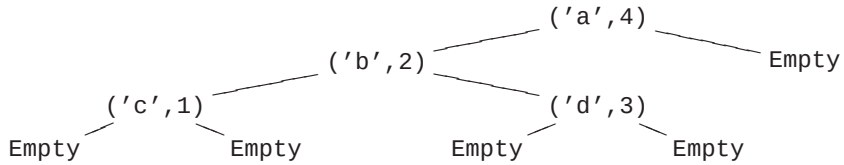
e que, como tal, satisfaz as propriedades seguintes:

$$rev \cdot cat = cat \cdot (rev \times rev) \cdot swap \tag{6}$$
$$rev \cdot singl = singl \tag{7}$$
$$rev \cdot \underline{Nil} = \underline{Nil} \tag{8}$$
$$rev \cdot rev = id \tag{9}$$

Demonstre (9) usando as outras propriedades acima enunciadas.

GRUPO III

Questão 7 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. Por exemplo, a árvore da Questão 4 será decorada da seguinte forma



Questão 8 Demonstre que o facto

$$do\,\{\ a \leftarrow x\ ;\ u(f\ a)\ \} \quad = \quad (\mathsf{F}\ f)\ x \tag{10}$$

é válido para toda a mónada F.
NB: recorde que a unidade u de F corresponde à função `return` do Haskell.
