Métodos de Programação I

2.º Ano da LMCC (7003N5) + LESI (5303O7)
Ano Lectivo de 2003/04

Exame (1.ª chamada da época normal) de Janeiro 2004
14h00
Salas 2201 a 2210

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)

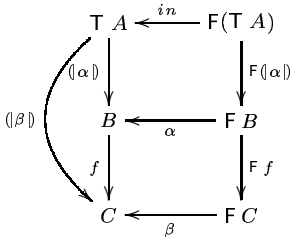
GRUPO I

Questão 1 Considere o seguinte raciocínio:

$$
\begin{aligned}
& k = [f, g] \Leftrightarrow \left\{ \begin{array}{l} k \cdot i_1 = f \\ k \cdot i_2 = g \end{array} \right. \\
\equiv & \quad \{ \ldots \text{(justifique)} \ldots \} \\
& h \cdot [i, j] = [f, g] \Leftrightarrow \left\{ \begin{array}{l} (h \cdot [i, j]) \cdot i_1 = f \\ (h \cdot [i, j]) \cdot i_2 = g \end{array} \right. \\
\equiv & \quad \{ \ldots \text{(justifique)} \ldots \} \\
& h \cdot [i, j] = [f, g] \Leftrightarrow \left\{ \begin{array}{l} h \cdot ([i, j] \cdot i_1) = f \\ h \cdot ([i, j] \cdot i_2) = g \end{array} \right. \\
\equiv & \quad \{ \ldots \text{(justifique)} \ldots \} \\
& h \cdot [i, j] = [f, g] \Leftrightarrow \left\{ \begin{array}{l} h \cdot i = f \\ h \cdot j = g \end{array} \right. \\
\equiv & \quad \{ \ldots \text{(justifique)} \ldots \} \\
& h \cdot [i, j] = [h \cdot i, h \cdot j]
\end{aligned}
$$

Identifique a propriedade da qual este raciocínio partiu? Justifique cada passo e indique qual das leis que constam do anexo foi deduzida.

Questão 2 Use a lei de fusão-cata



para provar a validade do seguinte facto, em Haskell: `x + foldr (+) y = foldr (+) (x+y)`.

Sugestão: interprete `foldr` como um catamorfismo de listas e faça $f = (x+)$.

Questão 3 O fragmento de código Haskell que se segue define parcialmente um tipo de dados estudado nesta disciplina:

```
inX = either Leaf Split

outX (Leaf a) = i1 a
outX (Split (t1,t2)) = i2 (t1,t2)

cataX a = a . (recX (cataX a)) . outX

hyloX a c = cataX a . anaX c
```

1. Identifique, definindo-o, o tipo de dados paramétrico `X`, e acrescente ao fragmento dado as definições, que faltam, de `recX` e `anaX`.
2. Identifique quais dos seguintes algoritmos se podem definir como hilomorfismos do tipo `X`, indicando, quanto aos outros, os tipos que desempenham função análoga (de estrutura de dados virtual):

FUNÇÃO	ALGORITMO	TIPO INTERMÉDIO
`mSort`	Merge sort	
`hanoi`	Torres de Hanoi	
`dfac`	Duplo factorial	
`qSort`	Quick sort	
`fac`	Factorial	
`fib`	Série de Fibonacci	
`iSort`	Odernação por inserção simples	

GRUPO II

Questão 4 Considere a seguinte definição *point-free* de uma função f:

$$f = [\mathsf{succ} \cdot \underline{0}, \mathsf{plus} \cdot \langle f \cdot \mathsf{pred}, f \cdot \mathsf{pred2} \rangle] \cdot \mathsf{zeroOrOne}?$$

em que

$$\mathsf{zeroOrOne}\ x = (x == 0) \mid\mid (x == 1)$$
$$\mathsf{plus} = \mathsf{uncurry}\ (+)$$
$$\mathsf{pred2} = \mathsf{pred} \cdot \mathsf{pred}$$

1. Escreva uma definição de f no estilo *pointwise*, justificando todos os passos para a sua obtenção.
2. Sejam

$$g_a = (\underline{()} + \langle \mathsf{pred}, \mathsf{pred2} \rangle) \cdot \mathsf{zeroOrOne}?$$
$$g_c = [succ \cdot \underline{0}, \mathsf{plus}]$$
$$h = (\!|g_c|\!) \cdot [\!(g_a)\!]$$

   Desenhe o diagrama correspondente ao hilomorfismo h e declare em Haskell o seu tipo intermédio. Qual o valor deste tipo correspondente ao cálculo de $h\ 4$?
3. Prove a equivalência $h = f$, justificando todos os passos.

Questão 5 O seguinte código diz respeito a uma função `label` que a partir de uma árvore de folhas de inteiros, produz uma outra árvore com a mesma forma, em que o conteúdo de cada folha é substituído pela sua soma com os conteúdos de todos as folhas à sua esquerda. Por exemplo, `label (Node (Leaf 1) (Node (Leaf 2) (Leaf 3)))` resulta em `Node (Leaf 1) (Node (Leaf 3) (Leaf 6))`. Assumindo que o tipo `Estado` foi já apropriadamente declarado como instância da classe Monad, complete o código em A e B.

```
data Estado a = E (Int -> (Int, a))

lab :: Tree Int -> Estado (Tree Int)
lab (Leaf x) = .............A.................
lab (Node esq dir) = do esq' <- lab esq
                        dir' <- lab dir
                        return (Node esq' dir')

label :: Tree a -> Tree Int
label t = ...........B..............
```
