Métodos de Programação I

2.º Ano da LMCC (7003N5) + LESI (5303O7)
Ano Lectivo de 1999/2000

Exame (época de recurso) — 4 de Setembro de 2000
14h30
Salas 2301 a 2303

NB: *Esta prova consta de **10** alíneas que valem, cada uma, 2 valores.*

1. *Responda a cada um dos grupos de questões em folhas separadas.*
2. *Para sua consulta, encontra anexa a esta prova a listagem das principais leis de programação estudadas na disciplina.*

PROVA SEM CONSULTA (3 horas)

GRUPO I

Questão 1 Considere o seguinte raciocínio:

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

Como se chama a propriedade de que o raciocínio partiu? Justifique cada passo e indique qual das leis que constam do anexo foi deduzida.

Questão 2 Prove a *lei de fusão* do condicional de McCarthy:

$$f \cdot (p \to g, h) \quad = \quad p \to f \cdot g, f \cdot h \tag{1}$$

Questão 3 Identifique quais das igualdades seguintes,

```
curry f a b = f(a,b)                                              (2)
curry g (f a) c = curry (g .\(a,c)->(f a, c)) a c                 (3)
```

expressas em sintaxe HASKELL, são propriedades válidas e identifique-as, desenhando o diagrama correspondente.

GRUPO II

Questão 4 Sejam dadas as seguintes funções, no contexto da biblioteca `Mpi.hs`:

```
f = either (const True) (not . snd)
g = either k (succ . fst)
```

onde `k` é uma função arbitrária. Identifique, justificando,

- o tipo de `[f,g]`, isto é, da lista contendo as funções `f` e `g`;
- o tipo de `(f,g)`, isto é, do par de funções `f` e `g`.

Sugestão: recorde o algoritmo de Hindley-Milner para inferência de tipos que foi estudado nas aulas práticas.

Questão 5 O fragmento de código HASKELL que se segue define parcialmente um componente de programação (módulo) HASKELL que estudou:

```
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. Complete a definição do operador functorial `map` associado ao tipo de dados `X`:

   ```
   instance Functor X
            where map f = ...
   ```

3. Identifique quais dos seguintes algoritmos se definem nesse módulo como hilomorfismos do tipo `X`, identificando, quanto aos outros, os módulos a que pertencem:

FUNÇÃO	ALGORITMO	MÓDULO (`*.hs`)
`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 III

Questão 6 Considere o seguinte algoritmo *standard* para conversão de um número inteiro (não negativo) para base 2, expresso sob a forma de uma *lista de zeros ou uns*:

```
base2 :: Int -> [Int]
base2 0 = []
base2 x = let (q,r) = divMod x 2
          in (base2 q)++[r]
```

Por exemplo, $base2\,11 = 1011$. De facto, se se correr `base2 11` em HUGS, obter-se-á `[1, 0, 1, 1]`.

1. É possível definir $base2$ com base num anamorfismo:

   $$base2 \quad \stackrel{\text{def}}{=} \quad reverse \cdot [\![(g)]\!]$$

   Identifique o gene g desse anamorfismo.

2. Seja agora $base10$ a inversa de $base2$, a função tal que

$$\begin{cases} base2 \cdot base10 = id \\ base10 \cdot base2 = id \end{cases}$$

se verificam. Construa $base10$ com base num catamorfismo, *i.é*, identifique h em

$$base10 \quad \stackrel{\text{def}}{=} \quad (\!|h|\!) \cdot reverse$$

Sugestão: atente no exemplo seguinte:

$$base10\,[1,0,1,1] \quad = \quad (((1 \times 2) + 0) \times 2) + 1) \times 2 + 1$$

Questão 7 Uma das primeiras linguagens de programação funcional foi o LISP, que apareceu há cerca de 40 anos. Nesta linguagem existe apenas um único suporte para representação de dados, designado por *expressão-S* — abreviatura de "expressão simbólica". Uma *expressão-S* ou é uma valor atómico ou é uma sequência (possivelmente vazia) de *expressões-S*. Considera-se um *átomo* toda a unidade de informação indivisível, não-estruturada (*i.é* "atómica").

Por exemplo, são átomos os inteiros e os 'strings' alfanuméricos, *e.g.* `10`, `-5`, `a12`, `xyz`. Dão-se a seguir exemplos de *expressões-S* não atómicas, escritas na própria sintaxe concreta do LISP:

```
()
(1)
(1 um 2 dois)
(1 (2 (3 (4))))
```

Seja

```
data SExp a = Atom a | Exp [ SExp a ]
```

a declaração de um tipo de dados em HASKELL para descrever *expressões-S*.

Desenhe o diagrama dos catamorfismos deste tipo e exprima a operação que conta o número de átomos presentes numa *expressão-S* como um desses catamorfismos.

Anexo–Cálculo de Funções

COMPOSIÇÃO

Natural-id	$f \cdot id = id \cdot f = f$	(4)
Associatividade	$(f \cdot g) \cdot h \quad = \quad f \cdot (g \cdot h)$	(5)

PRODUTO

Universal-$\times$	$k = \langle f, g \rangle \quad \Leftrightarrow \quad \begin{cases} \pi_1 \cdot k = f \\ \pi_2 \cdot k = g \end{cases}$	(6)
Cancelamento-$\times$	$\pi_1 \cdot \langle f, g \rangle = f \quad , \quad \pi_2 \cdot \langle f, g \rangle = g$	(7)
Reflexão-$\times$	$\langle \pi_1, \pi_2 \rangle \quad = \quad id_{A \times B}$	(8)
Fusão-$\times$	$\langle g, h \rangle \cdot f = \langle g \cdot f, h \cdot f \rangle$	(9)
Absorção-$\times$	$(i \times j) \cdot \langle g, h \rangle = \langle i \cdot g, j \cdot h \rangle$	(10)
Functor-$\times$	$(g \cdot h) \times (i \cdot j) \quad = \quad (g \times i) \cdot (h \times j)$	(11)
Functor-id-$\times$	$id_A \times id_B \quad = \quad id_{A \times B}$	(12)

COPRODUTO

Universal-$+$	$k = [f \, , g] \quad \Leftrightarrow \quad \begin{cases} k \cdot i_1 = f \\ k \cdot i_2 = g \end{cases}$	(13)
Cancelamento-$+$	$[g \, , h] \cdot i_1 = g \quad , \quad [g \, , h] \cdot i_2 = h$	(14)
Reflexão-$+$	$[i_1 \, , i_2] \quad = \quad id_{A+B}$	(15)
Fusão-$+$	$f \cdot [g \, , h] = [f \cdot g \, , f \cdot h]$	(16)
Absorção-$+$	$[g \, , h] \cdot (i + j) = [g \cdot i \, , h \cdot j]$	(17)
Functor-$+$	$(g \cdot h) + (i \cdot j) = (g + i) \cdot (h + j)$	(18)
Functor-id-$+$	$id_A + id_B = id_{A+B}$	(19)

EXPONENCIAÇÃO

Universal	$k = \overline{f} \quad \Leftrightarrow \quad f = ap \cdot (k \times id)$	(20)
Cancelamento	$f = ap \cdot (\overline{f} \times id)$	(21)
Reflexão	$\overline{ap} = id_{B^A}$	(22)
Fusão	$\overline{g \cdot (f \times id)} = \overline{g} \cdot f$	(23)
Absorção	$\overline{f \cdot g} = f^A \cdot \overline{g}$	(24)
Functor	$(g \cdot h)^A = g^A \cdot h^A$	(25)
Functor-id	$id^A = id$	(26)

MISC.

Lei da troca	$[\langle f, g \rangle \, , \langle h, k \rangle] \quad = \quad \langle [f \, , h], [g \, , k] \rangle$	(27)