Métodos de Programação I

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

Exame (2.ª chamada da época normal) — 31 de Janeiro 2004
9h30
Salas 2105, 2111

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 A seguinte função calcula a $n+1$-ésima linha do triângulo de Pascal

```
pascal 0 = [1]
pascal (n+1) = 1:(soma (pn,(tail pn)))
        where pn = pascal n

soma ([],x) = x
soma (x,[]) = x
soma ((a:as),(b:bs)) = (a+b):(soma (as,bs))
```

Por exemplo, `pascal 0` é a lista `[1]`, `pascal 2` é a lista `[1, 2, 1]`, *etc.*, *cf.*

$$
\begin{array}{ccccccccccc}
 & & & & & 1 & & & & & \\
 & & & & 1 & & 1 & & & & \\
 & & & 1 & & 2 & & 1 & & & \\
 & & 1 & & 3 & & 3 & & 1 & & \\
 & 1 & & 4 & & 6 & & 4 & & 1 & \\
1 & & 5 & & 10 & & 10 & & 5 & & 1
\end{array}
$$

1. Indique o tipo da função `soma` e exprima-a como um anamorfismo.
2. Exprima a função `pascal` como um hilomorfismo e apresente o diagrama correspondente.

Questão 2 Considere a função definida como

$$
\begin{array}{ll}
\mathsf{media} & = (\mathsf{uncurry}(/)) \circ \mathsf{aux} \\
\mathsf{where} & \mathsf{aux} = \langle (\!|[\underline{0}, \mathsf{uncurry}\ (+)]|\!), \mathsf{length} \rangle
\end{array}
$$

1. Defina aux como um catamorfismo.
2. Poderá a função media ser definida como um catamorfismo? Justifique a sua resposta.

GRUPO II

Questão 3 Uma das operações conhecidas sobre listas é a da inversão:

```
invl [] = []
invl (a:l) = (invl l) ++ [a]
```

1. Calcule a definição de $invl$, dada acima em Haskell, a partir do seguinte catamorfismo:

$$
invl \stackrel{\mathrm{def}}{=} (\!|\underbrace{[nil, uconc \cdot swap \cdot (singl \times id)]}_{g}|\!) \tag{1}
$$

onde $nil = \underline{[\,]}$ e $uconc = uncurry(++)$, apoiando a sua resposta por um diagrama explicativo.

2. Converta para notação com variáveis a propriedade

$$invl \cdot uconc \quad = \quad uconc \cdot (invl \times invl) \cdot swap \tag{2}$$

e complete as igualdades seguintes por forma a exprimirem também propriedades válidas:

$$invl \cdot singl \quad = \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \tag{3}$$

$$uconc \cdot (\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ ) \quad = \quad cons \tag{4}$$

em que $cons(a, l) = a : l$.

3. Complete as justificações da seguinte prova da propriedade involutiva de $invl$:

$$
\begin{array}{ll}
& invl \cdot invl = id \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& invl \cdot (\!|g|\!) = (\!|in|\!) \\
\Leftarrow & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& invl \cdot g = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \text{ expansão de } g \text{ (1) } \} \\
& invl \cdot [nil, uconc \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& [invl \cdot nil, invl \cdot uconc \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& [nil, uconc \cdot (invl \times invl) \cdot swap \cdot swap \cdot (singl \times id)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& [nil, uconc \cdot (invl \cdot singl \times invl)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \text{ pela lei (3)} \} \\
& [nil, uconc \cdot (singl \times invl)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& [nil, cons \cdot (id \times invl)] = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& [nil, cons] \cdot (id + id \times invl) = in \cdot (id + id \times invl) \\
\equiv & \quad \{ \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \ \} \\
& \mathrm{T}
\end{array}
$$

Questão 4 Se pedir ao GHC informações sobre a class `Monad`,

```
Prelude> :i Monad
```

obterá

```
-- Monad is a class
class Monad m :: (* -> *) where {
    (>>=) :: forall a b. m a -> (a -> m b) -> m b;
    (>>) :: forall a b. m a -> m b -> m b {- has default method -};
    return :: forall a. a -> m a;
    fail :: forall a. String -> m a {- has default method -};
    }
```

Idenfifique qual das funções disponibilizadas por esta classe corresponde à seguinte função f, em Haskell

```
f x y = (x >>= return) >>= (const y)
```

Sugestão: Poderá ser-lhe útil recordar das aulas teóricas, o diagrama

$$
\begin{tikzcd}
\mathsf{F}^2\, A \arrow[d, "\mu"'] & \mathsf{F}\, A \arrow[l, "u"'] \arrow[dl, "id"] \arrow[d, "\mathsf{F}\, u"] \\
\mathsf{F}\, A & \mathsf{F}^2\, A \arrow[l, "\mu"]
\end{tikzcd}
$$
