Métodos Formais de Programação I +
Opção - Métodos Formais de Programação I

4.º Ano da LMCC (7007N2) + LESI (5307P6)
Ano Lectivo de 2001/02

Exame (época de recurso) — 6 de Setembro 2002
14h30
Sala 1103

NB: *Esta prova consta de **7** alíneas todas com a mesma cotação*

PROVA SEM CONSULTA (2 horas)

Questão 1 Considere a seguinte especificação em VDM-SL:

```
seq2ff[@A] : seq of @A -> map nat to @A
--
-- seq2ff(l) converts sequence l into a finite
-- mapping keeping track of the original
-- element positions.
--
seq2ff(l) == { i |-> l(i) | i in set inds l };
```

1. Indique, justificando, qual o resultado que o interpretador das VDMTOOLS fornece para o cálculo das expressões seguintes, onde `seq2ff` ocorre:

```
dom seq2ff[char](tl "abc")        (1)
seq2ff[char](hd [])               (2)
dom seq2ff([])                    (3)
```

2. Podendo a função `seq2ff` ser especificada por indução sobre listas, alguém propôs a seguinte definição:

```
seq2ff(l) ==
  cases l:
     []      -> {|->},
     [h] ^ t -> seq2ff[@A](t) munion {len t + 1 |-> h }
  end;
```

Estará esta proposta correcta? Justifique a sua resposta.

Questão 2 Considere o tipo de dados *árvore binária de inteiros com folhas*

```
LTree =  Leaf | Node ;
Leaf  :: value: int ;
Node  :: left: LTree right: LTree ;
```

especificado em VDM-SL, bem como as funções

```
all1: LTree -> LTree
all1(t) ==
   cases t :
      mk_Leaf(-) -> mk_Leaf(1) ,
      mk_Node(t1,t2) -> mk_Node(all1(t1),all1(t2))
   end;

add: LTree -> int
add(t) ==
   cases t :
      mk_Leaf(i) -> i ,
      mk_Node(t1,t2) -> add(t1) + add(t2)
   end;
```

1. Identifique a função `f(t) == add(all1(t))` completando o processo de cálculo que se segue e convertendo o catamorfismo resultante para VDM-SL com variáveis:

$$
\begin{array}{ll}
 & add \cdot all1 \\
= & \quad \{ \ \ldots\ldots \} \\
 & (\!|[id, +]|\!) \cdot all1 \\
= & \quad \{ \ \ldots\ldots \} \\
 & (\!|[id, +]|\!) \cdot (\!|in \cdot (\underline{1} + id \times id)|\!) \\
= & \quad \{ \ \ldots\ldots \} \\
 & (\!|[id, +] \cdot (\underline{1} + id \times id)|\!) \\
= & \quad \{ \ \ldots\ldots \} \\
 & (\!|[\underline{1}, + \cdot (id \times id)]|\!) \\
= & \quad \{ \ \ldots\ldots \} \\
 & (\!|[\underline{1}, +]|\!) \\
= & \quad \{ \ \ldots\ldots \} \\
 & f
\end{array}
$$

2. Se a alínea anterior tivesse sido baseada no tipo `seq1 of int` em lugar de `LTree`, à função `f` corresponderia uma função bem conhecida da notação VDM-SL. Qual? Justifique a sua resposta, incluindo um diagrama explicativo.

3. Que função é realizada compondo a função

```
g : seq1 of int -> LTree
g(l) ==
   cases l :
     [e] -> mk_Leaf(e),
     others -> let l1 ^ l2 in set {l}
               be st
                   abs (len l1 - len l2) < 2
               in
                   mk_Node(g(l1),g(l2))
   end;
```

   com a função `add` da alínea 1? Justifique a sua resposta escrevendo directamente em VDM-SL a fusão das duas funções.

Questão 3 Recorde o *Problema 5* das práticas laboratoriais desta disciplina, em que se considerou o seguinte modelo VDM-SL para especificar um sistema de gestão de um "banco brinquedo":

```
BAMS = map AccId to Account;
Account :: H: set of AccHolder
           B: Amount;

AccId     = seq of char;
AccHolder = seq of char;
Amount = int;
```

A seguinte especificação da operação de crédito numa conta,

```
Credit: BAMS * AccId * Amount -> BAMS
Credit(bams,n,m) ==
     selUp[AccId,Account]
          ({n})
          (lambda x:Account & mk_Account(x.H, x.B + m))
          (bams);
-- pre n in set dom bams;
```

baseia-se no operador genérico de “actualização selectiva” de funções finitas

```
selUp[@A,@B] : set of @A -> (@B -> @B) -> map @A to @B -> map @A to @B
selUp(s)(f)(x) == x ++ ffmap[@A,@B,@B](f)(s <: x);
```

que por sua vez se baseia no operador functorial

```
ffmap[@A,@B,@C]: (@B -> @C) -> map @A to @B -> map @A to @C
ffmap(f)(x) == { k |-> f(x(k)) | k in set dom x };
```

1. Note que a pre-condição de `Credit` foi posta em comentário. Qual o comportamento de `Credit` para a situação em que tal predicado se não verifica? Justifique.
2. Mostre que

$$\texttt{ffmap}\, f \quad = \quad \{|in \cdot (id + (id \times f) \times id)|\}$$

para $in = [empty, join]$, onde

```
empty() == {|->}
join(mk_(a,b),m) == m munion { a |-> b };
```

NB: apoie a sua resposta num diagrama explicativo.
