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

4.º Ano da LMCC (7007N2) + LESI (5307P6)
Ano Lectivo de 2000/01

Exame (2.ª chamada) — 24 de Janeiro de 2001
09h30
Sala 2312

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

PROVA SEM CONSULTA ($3\ horas$)

Questão 1 Responda às seguintes alíneas:

1. Complete as "..." da seguinte definição, em VDM-SL, de um combinador funcional bem seu conhecido:

```
foldr[@A,@B] : ............. * @B -> seq of @A -> ...
foldr(f,u)(l) == if l=[] then u
                         else ..................... ;
```

2. Pretende-se agora adaptar a solução da alínea anterior à definição do combinador equivalente para 'folding' sobre conjuntos. Se designarmos esse combinador por `setFold`, deverá ter-se

$$setFold(f,u) \stackrel{\text{def}}{=} \{\!|[\underline{u}, f]|\!\} \tag{1}$$

Complete então a definição

```
setFold[@A,@B] : (@A*@B -> @B) * @B -> set of @A -> @B
setFold(f,u)(s) == ................................ ;
```

3. Identifique, em cada uma das equações que se seguem, as funções f aí presentes, por forma à equação ser válida quaisquer que sejam as outras variáveis em jogo:

$$foldr(f,\emptyset)(l) = elems\ l \tag{2}$$
$$setFold(f,\emptyset)(s) = s \tag{3}$$
$$setFold(f,\emptyset)(s) = \mathcal{P}g \tag{4}$$

Justifique invocando definições ou propriedades que conhece.

Questão 2 Um algoritmo conhecido para compressão de imagens, representadas por sequências de 'bytes', redu-las a sequências de intervalos, omitindo todos os 'bytes' entre os extremos de uma série de valores consecutivos. Exemplificando, a imagem representada por

$$[4, 5, 6, 10, 20, 21, 22, 23, 24]$$

será reduzida a

$$[(4, 6), (10, 10), (20, 24)]$$

Interprete a seguinte especificação em VDM-SL de uma função que realiza tal compressão:

```
compress : seq of int -> seq of (int*int)
compress(l) == foldr[int,seq of (int*int)](g,[])(l) ;

g : int * seq of (int*int) -> seq of (int*int)
g(a,l)  == if l = []
           then [mk_(a,a)]
           else let mk_(b,c) = hd l,
                    i = if b = a + 1 then [mk_(a,c)]
                                     else [mk_(a,a),mk_(b,c)]
                in i ^ tl l ;
```

1. Indique (justificando) o resultado que deverá obter no interpretador de VDMTOOLS quando invoca

   ```
   vdm> p compress([9,8,7,2,3])
   ```

2. Partindo da análise do resultado da alínea anterior, especifique uma *pré-condição* necessária ao bom funcionamento de `compress`.
3. Verifica-se que `compress` garante uma propriedade à saída quando aplicada a argumentos que validem essa pré-condição. Especifique essa propriedade.

Questão 3 Recorde o *Problema 2* das sessões laboratoriais desta disciplina, baseado no tipo

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

1. São dadas a seguir duas definições alternativas da operação de abertura de uma conta,

   ```
   OpenAccount(bams,n,h,m) ==
         bams munion { n |-> mk_Account(h,m) }
   pre not n in set dom bams;
   ```

   e

   ```
   OpenAccount(bams,n,h,m) ==
         { n |-> mk_Account(h,m) } ++ bams ;
   ```

   Suponha que, ao interpretar este modelo em VDMTOOLS, com a opção "`Enable check of pre-conditions`" activada, testa a situação em que se tenta abrir uma conta que já existe. Compare o comportamento destas duas alternativas para essa situação.
2. Especifique a operação que calcula o montante total de dinheiro armazenado no banco.
3. Especifique uma outra operação que indique todas as contas registadas para um determinado cliente.

Questão 4 Recorde o *Problema 3* das sessões laboratoriais desta disciplina, onde se abordou a operação

```
mseCup[@A] : map @A to nat * map @A to nat -> map @A to nat
```

Partindo da definição formal deste operador que foi estudada nas aulas, justifique a sua opinião sobre a validade das propriedades que se seguem:

$$m = mseCup[@A](m, \{\mid \text{->} \}) \quad (5)$$

$$mseCup[@A](m, m) = m \quad (6)$$
