Exame de Recurso de

Análise e Desenvolvimento de Algoritmos

Data: 11 de Julho de 2007 Duração: 2h 30m

Em cada grupo, leia cuidadosamente o enunciado de todas as alíneas, antes de começar a responder.

1. Com base na Axiomática de Hoare, verifique formalmente a correcção parcial dos algoritmos seguintes:

   (a) $\{a, b \in \mathbb{N}\}$

```
x := 0; c := 0;
while c < a do
      begin c := c + 1;
            x := x + b;
      end;
```

   $\{x = a \times b\}$

   (b) $\{a, b \in \mathbb{N}\}$

```
x := 0; c := a;
while c > 0 do
      begin c := c - 1;
            x := x + b;
      end;
```

   $\{x = a \times b\}$

   (c) $\{a, b \in \mathbb{N}\}$

```
x := a; y := b; s := 0;
while y > 0 do
      if not odd(y)
      then begin x := x * 2;
                 y := y div 2
           end;
      else begin s := s + x;
                 x := x * 2;
                 y := (y-1) div 2
           end;
```

   $\{s = a \times b\}$

   (d) Qual é o número mínimo de somas realizadas por este último algoritmo? Em que condições ocorre? E o número máximo de somas?

2. Seja $x[1..n]$ um vector de elementos inteiros. Dizemos que o vector está **ordenado**, quando os seus elementos obedecem à propriedade:

$$\{ \forall\, i \in [1..n-1],\ x_i \leq x_{i+1} \}$$

   (a) Construa uma função iterativa, com resultado do tipo lógico:

   ```
   function ordenado(x:vector; n:indice): boolean;
   ```

   para verificar **se** um dado vector está ou não **ordenado**.

   (b) Elabore uma versão recorrente da função anterior.

   (c) Para cada um dos algoritmos anteriores, calcule o número de comparações efectuadas entre elementos do vector. O que pode concluir acerca dos tempos de execução previstos para as duas funções?

   (d) Construa um sub-programa para calcular o comprimento do maior (mais longo) sub-vector ordenado de um dado vector.

3. Considere o Tipo Abstracto de Dados *Natural*, destinado a manipular números naturais de grandes dimensões. A representação mais simples consiste numa Lista Ligada Linear, com um dígito por nó, onde o primeiro elemento contém o algarismo das unidades. Assuma que está já implementado o módulo:

   ```
   function copia(a : natural) : natural;
   ```
   $\{:= a\}$

   (a) Construa uma função para calcular a soma de dois *Naturais*:

   ```
   function soma(a, b : natural) : natural;
   ```
   $\{a + b\}$

   (b) Construa uma função para comparar dois *Naturais*:

   ```
   function menor(a, b : natural) : boolean;
   ```
   $\{a < b\}$

   (c) Utilizado os módulos anteriores e o algoritmo de **1.(a)**, escreva uma função para calcular o produto de dois *Naturais*.

   ```
   function produto(a, b : natural) : natural;
   ```
   $\{a \times b\}$

4. Por fim considere o Tipo Abstracto de Dados *Binario*, semelhante ao TAD *Natural*, mas onde os números naturais estão representados em **Notação Binária**. Escreva um módulo para calcular o produto de dois *Binarios*, utilizando o algoritmo de **1.(c)**.