Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI 2611 Algoritmos 1

Tema: Demostración de la correctitud de llamadas a procedimientos y modificaciones de arreglos.

TAREA 3: ejercicios 3a), 10, 14, 15, 17, 18

1. Dado el encabezado de procedimiento siguiente:

   proc duplicar(**entrada** e:entero; **salida** s:entero)

   {Pre: $e \geq 0$}

   {Post: $s = 2*e$ }

   Muestre que las siguientes tripletas son correctas:

   a) { $n \geq 0$ } duplicar(n,m) { $m = 2*n$ }
   b) {true} duplicar(n*n,m) {$m= 2*n*n$ }
   c) { $n \geq -5$ } duplicar(n+10,m) {$m \bmod 2 = 0$ }
   d) { $n \geq 5$ } duplicar(n,m) { $m \geq 10$ }
   e) { $a > 7$ } duplicar(a + b*b, c) { $c \geq 16$ }
   f) { $10 < m < n$ } duplicar(n,m) { $m \geq 24$ }
   g) { $n \geq 5$ y $m \geq 5$ } duplicar(n,m) { $m > 0$ }
   h) { $n \geq 5$ } duplicar(n,n) { $n \geq 10$ }
   i) { $a > 7$ } duplicar(a + b*b, a) { $a \geq 16$ }
   j) { $a > 7$ } duplicar(a + b*b, b) { $b \geq 16$ }
   k) { $0 < k \leq N \wedge N-k = X$ } duplicar(k, k) { $N-k < X$ }

   Si el encabezado del procedimiento duplicar fuese**:**

   proc duplicar(**entrada-salida** num:entero)

   {Pre: $num \geq 0$}

   {Post: $num = 2* num_0$, donde num_0 es el valor inicial de num}

   Muestre que las siguientes tripletas son correctas:

   l) { $n \geq 5$ } duplicar(n) { $n \geq 10$ }
   m) { $0 < k \leq N \wedge N-k = X$ } duplicar(k) { $N-k < X$}
   n) { $p \geq 0 \wedge p \bmod 2 = 0$ } duplicar(p) { $p \bmod 4 = 0$ }
   o) { $p \geq 0 \wedge p \bmod 2 = 1$ } duplicar(p) { $p \bmod 4 = 2$ }

2. Suponga que cuenta con procedimiento de exponenciación:

   proc exponenciar(entrada base, exp: int; salida result:int)
   { Pre: $exp \geq 0$ }
   { Post: $result = base^{exp}$ }

   y se tiene un problema computacional especificado como sigue:

   [const dato0, dato1, dato2, dato3, dato4, dato5: int;
   var resA, resB, resC : int;

{ dato1 ≥ 0 ∧ dato3 ≥ 0 ∧ dato5 ≥ 0 }
TresExp
{ resA = $dato0^{dato1}$ ∧ resB = $dato2^{dato3}$ ∧ resC = $dato4^{dato5}$ }
]

Demuestre que el problema TresExp se puede resolver con la instrucción:

```
exponenciar(dato0, dato1, resA);
exponenciar(dato2, dato3, resB);
exponenciar(dato4, dato5, resC)
```

Recuerde que para demostrar cada secuenciación (;) hacen falta aserciones intermedias, que a la vez serán usadas en la demostración de correctitud de cada llamada. Estas deben ser propuestas por intuición, salvo que sean calculadas por wp (precondición más débil). Use como primera aserción intermedia a:

$$\{ resA = dato0^{dato1} \wedge dato3 \geq 0 \wedge dato5 \geq 0 \}$$

sugiera usted su propia segunda aserción intermedia y complete la demostración de correctitud.

3. Usando el mismo procedimiento exponenciar del ejercicio anterior, se quiere resolver el siguiente problema:

```
[ const a, b, c : int;
  var r : int;
   { b ≥ 0 ∧ c ≥ 0 }
   ExpSobreExp
   { r =  a^(b^c)}
]
```

Donde **a^(b^c)** es el resultado de elevar **a** a la potencia **b^c** (**b** a la potencia **c**).

Demuestre la correctitud de las siguientes dos soluciones al problema ExpSobreExp:

a.
```
var temp : int;
exponenciar(b,c,temp);
exponenciar(a,temp,r);
```

b.
```
exponenciar(b,c,r);
exponenciar(a,r,r);
```

4. Con la misma definición de encabezado para exponenciar del ejercicio anterior, demuestre la correctitud del siguiente programa:

```
[  const N : int;
     a : array [0..N) of int;
  var r : int;
  { N > 0 }
  [ var k:int;
    r, k := a[0], 1;
    { Invariante: 1 ≤ k ≤ N ∧ r = (max i: 0 ≤  i < k : a[i]^(i+1) )}
    do k != N →
      [  var temp : int;
         exponenciar(a[k],k+1,temp);
```

```
        r := r max temp;
        k := k + 1;
      ]
    od
  ]
  { r = (max i: 0 ≤ i < N : a[i]^(i+1) ) }
]
```

5. Construya otra solución iterativa para el mismo problema de la pregunta anterior, pero usando el invariante correspondiente a una iteración "descendente", esto es, el invariante:

   $0 \le k \le N-1 \wedge r = (\max i: k \le i < N : a[i]^{(i+1)})$

   Demuestre la correctitud de su solución.

6. La regla A-2 (de la guía) resuelve dos problemas de la regla A-1 tal como fue explicado en el texto: usar información proveniente del estado inicial en el estado final, y permitir solapamiento entre los parámetros reales de entrada y los parámetros reales de salida.

   El primer problema ha podido tratarse por separado, aunque en el texto se resolvió resolverlo en conjunto con el segundo. Enuncie usted una regla A-1' en la que se resuelve sólo el primer problema pero no el segundo: esto es, aún se prohibirá ocurrencias del parámetro real de salida en el parámetro real de entrada, pero Pllam será combinado con Qdef.

7. Muestre que con su regla A-1' del ejercicio anterior, se puede demostrar la correctitud de:

   - Las cuatro llamadas:
   { n ≥ 5 } duplicar(n,m) { m ≥ 10 }
   { a > 7 } duplicar(a + b*b, c) { c ≥ 16 }
   { 10 < m < n } duplicar(n,m) { m ≥ 24 }
   { n ≥ 5 y m ≥ 5 } duplicar(n,m) { m > 0 }

   Donde:

   proc duplicar(**entrada** e:entero; **salida** s:entero)

   {Pre: e ≥ 0}

   {Post: s = 2*e }

8. Suponga que cuenta con un nuevo procedimiento de exponenciación, definido como sigue:

   proc exponenciar(entrada exp: int; entrada-salida num:int)
   { Pre: exp ≥ 0 }
   { Post: num = num_0^{exp} }

   Con este nuevo procedimiento, el problema dado por la especificación:

   [const dato0, dato1, dato2, dato3, dato4, dato5: int;
   var resA, resB, resC : int;

{ dato1 ≥ 0 ∧ dato3 ≥ 0 ∧ dato5 ≥ 0 }
TresExp
{ resA = $dato0^{dato1}$ ∧ resB = $dato2^{dato3}$ ∧ resC = $dato4^{dato5}$ }
]

Se puede resolver con la instrucción:

```
resA := dato0; exponenciar(dato1, resA);
resB := dato2; exponenciar(dato3, resB);
resC := dato4; exponenciar(dato5, resC)
```

Demuestre la correctitud de esta solución. Recuerde que para demostrar cada secuenciación (;) hacen falta aserciones intermedias, que a la vez serán usadas en la demostración de correctitud de cada llamada. Estas deben ser propuestas por intuición, salvo que sean calculadas por wp (precondición más débil). Sugerimos que use como primera aserción intermedia a:

{ resA = dato0 ∧ dato0≥ 0∧ dato3≥ 0 ∧ dato5 ≥ 0 }

y como la segunda a:

{ resA = $dato0^{dato1}$ ∧ dato3≥ 0 ∧ dato5 ≥ 0 }

Proponga usted las aserciones intermedias restantes y complete la demostración de correctitud.

9. Resolver:

9.1. Complete adecuadamente en el siguiente programa la postcondición y el invariante:

```
[ const a, b, c : int;
  var r : int;
  { b ≥ 0 ∧ c ≥ 1 }
  r := a;
  exponenciar(b, r);
  [
    var k : int;
    k := 1;
    { Invariante: 1 ≤ k ≤ χ ∧ r = ???
      función de cota : c-k}
    do k != c →
        exponenciar(b,r);
        k := k + 1;
    od
  ]
  { r = ????  }
]
```

9.2. Muestre que el problema resuelto por el programa anterior ha podido ser resuelto por un programa compuesto sólo por una asignación y una llamada.

10. Utilizando una función:

```
func exponenciacion(b, e : int) → int
      { Pre: e ≥ 0 }
```

{ Post: exponenciación = b^e }

demuestre que los problemas TresExp y ExpSobreExp de los ejercicios (2) y (3), pueden ser resueltos correctamente por las instrucciones:

```
resA := exponenciación(dato0, dato1);
resB := exponenciación(dato2, dato3);
resC := exponenciación(dato4, dato5)
```

y

```
r := exponenciación(a, exponenciación(b, c))
```

respectivamente.

11. Desarrolle una función que calcule el número de segundos correspondientes a una duración expresada en días, horas, minutos y segundos. Utilice dicha función para escribir un programa que dadas tres duraciones expresadas en días, horas, minutos y segundos, calcule la cantidad total de segundos de la suma de esas tres duraciones. Haga las demostraciones de correctitud que correspondan.

12. Motrar la correctitud de la siguiente terna de Hoare:

    {x>0} P(x) {x mod 2 = 1} donde se tiene que

    proc P(**in-out** a:entero)
    {pre: $a \geq 0$} {post: $a=2*a_0+1$}

13. Motrar la correctitud de la siguiente terna de Hoare:
    {$n \geq m \wedge m \geq 0$} Pfactorial(n,c) {$c=(\prod i: 1 \leq i \leq n : i)$ donde se tiene que

    proc Pfactorial(**in** x: entero, **out** f:entero)
    {pre: $x \geq 0$} {post: $f = (\prod i: 1 \leq i \leq x : i)$}

14. Motrar la correctitud de la siguiente terna de Hoare:

    {$0 \leq k \leq N \wedge s=(\sum i: 0 \leq i < k : (\prod i: 0 \leq i < k : x^i)/(\prod i: 1 \leq i < k : i))$}

    s,k:= s+potencia(x,k)/factorial(k),k+1

    { $s=(\sum i: 1 \leq i \leq k : (\prod i: 0 \leq i \leq k : x^i)/(\prod i: 1 \leq i \leq k : i))$ }

    donde se tiene que

    func potencia(y: real,n:entero) -> real
    {pre: $n \geq 0$} {post: potencia = $(\prod i: 0 \leq i < n : y^i)$}

    func factorial(x: entero) -> entero
    {pre: $x \geq 0$} {post: factorial = $(\prod i: 1 \leq i \leq x : i)$}

15. Motrar la correctitud de la siguiente terna de Hoare:

{ A[i]=X ∧ A[j]=Y } A[i]:= A[i]+A[j] { A[i]=X+Y }

16. Motrar la correctitud de la siguiente terna de Hoare:

{(∀i: 0≤i<k: A[i] = 2^i) ∧ 0≤k≤N ∧ k≠N}
A[k] := 2^k
{(∀i: 0≤i<k: A[i] = 2^i) ∧ 0≤k≤N }

17. Motrar la correctitud de la siguiente terna de Hoare:

{(∀i: 0≤i<k: S[i]=V[i]*V[i]) ∧ 0≤k≤N ∧ k≠N}
S[k],k := V[k]*V[k], k+1
{(∀i: 0≤i<k: S[i]=V[i]*V[i]) ∧ 0≤k≤N}

18. Motrar la correctitud de la siguiente terna de Hoare:

{i≥0 ∧ i<N-1 ∧ A[i]=X ∧ A[i+1]=Y} Intercambio (N,A,i) {A[i]=Y ∧ A[i+1]=X}

donde se tiene que:

proc Intercambio (**in** M: entero, **in-out** A:arreglo [0..M) de enteros,
in r:entero)
{pre: 0≤r<M-1} {post: A[r]=A_0[r+1] ∧ A[r+1]=A_0[r]}

19. Resuelva los siguientes problemas mediante algoritmos y demuestre su correctitud.

a) Dado un arreglo `A` de enteros sustituya todas las ocurrencias del valor `x` por el valor `y`
b) Dado un arreglo `A` ordenado insertar un elemento `e` en la posición que corresponda para que mantenga el orden.
c) Compactar un arreglo `A` para que todos los elementos diferentes de cero estén en la parte inicial. Ejemplo, si `A = <1, 0, 3, 0, 2, 4, 0, 0, 3, 7, 0>` el resultado sería `A = < 1, 3, 2, 4, 3, 7, 0, 0, 0, 0, 0>.`
d) Dados dos arreglos `A` y `B` de tamaño `N`, encontrar el vector `C` que resulta de la mezcla ordenada de los dos arreglos. `A` y `B` están ordenados no decrecientemente.