



Comentarios:

Hola, este no es un resuelto oficial, tiene el logo del DC porque me parecio divertido copiar el formato de la guia.

Ejercicio 1. ★ Las siguientes especificaciones no son correctas. Indicar por qué, y corregirlas para que describan correctamente el problema.

- a) **buscar:** Dada una secuencia y un elemento, devuelve en *result* la posición de la secuencia en la cual se encuentra el elemento.

```
proc buscar (in l: seq( $\mathbb{R}$ ), in elem:  $\mathbb{R}$ , out result:  $\mathbb{Z}$ ) {  
  Pre {elem  $\in$  l}  
  Post {l[result] = elem}  
}
```

- b) **progresionGeometricaFactor2:** Indica si la secuencia *l* representa una progresión geométrica factor 2. Es decir, si cada elemento de la secuencia es el doble del elemento anterior.

```
proc progresionGeometricaFactor2 (in l: seq( $\mathbb{R}$ ), out result: Bool {  
  Pre {True}  
  Post {result = True  $\leftrightarrow$  (( $\forall i : \mathbb{Z}$ )( $0 \leq i < |l| \rightarrow l[i] = 2 * l[i - 1]$ ))}  
}
```

- c) **minimo:** Devuelve en *result* el menor elemento de *l*.

```
proc minimo (in l: seq( $\mathbb{R}$ ), out result:  $\mathbb{Z}$ ) {  
  Pre {True}  
  Post {( $\forall y : \mathbb{Z}$ )( $(y \in l \wedge y \neq x) \rightarrow y > result$ )}  
}
```

Respuesta

- a) La **Pre** no aclara que pasa cuando hay mas de una aparición de *elem* en *l*, y hace falta pedir que *result* este en el rango de *l*.

```
proc buscar (in l: seq( $\mathbb{R}$ ), in elem:  $\mathbb{R}$ , out result:  $\mathbb{Z}$ ) {  
  Pre {elem  $\in$  l  $\wedge$  cantApariciones(elem, l) = 1}  
  Post { $0 \leq result < |l| \wedge l[result] = elem$ }  
}
```

- b) Este es más facil de ver que el anterior, cuando $i = 0$, va a tratar de acceder a la posición $l[0 - 1]$, que es cualquier cosa. Y creo que crashearia con una lista vacia o de un elemento.

```
proc progresionGeometricaFactor2 (in l: seq( $\mathbb{R}$ ), out result: Bool {  
  Pre {True}  
  Post {result = True  $\leftrightarrow$  (( $\forall i : \mathbb{Z}$ )( $0 \leq i < |l| - 1 \rightarrow 2 * l[i] = l[i + 1]$ ))}  
}
```

- c) No se para que esta ese $y \neq x$, y tendria que haber pedido en la **Pre** que *result* pertenezca a *l*.

```
proc minimo (in l: seq( $\mathbb{R}$ ), out result:  $\mathbb{Z}$ ) {  
  Pre {result  $\in$  l}  
  Post {( $\forall y : \mathbb{Z}$ )( $y \in l \rightarrow y > result$ )}  
}
```

Ejercicio 2. La siguiente no es una especificación válida, ya que para ciertos valores de entrada que cumplen la precondición, no existe una salida que cumpla con la postcondición.

```

proc elementosQueSumen (in  $l : seq\langle\mathbb{Z}\rangle$ , in suma:  $\mathbb{Z}$ , out result :  $seq\langle\mathbb{Z}\rangle$ ) {
  Pre {True}
  Post {
    /* La secuencia result está incluida en la secuencia l */
     $(\forall x : \mathbb{Z})(x \in result \rightarrow \#apariciones(x, result) \leq \#apariciones(x, l))$ 
    /* La suma de la result coincide con el valor de la suma */
     $\wedge suma = \sum_{i=0}^{|result|-1} result[i]$ 
  }
}

```

- Mostrar valores para l y $suma$ que hagan verdadera la precondición, pero tales que no exista $result$ que cumpla la postcondición.
- Supongamos que agregamos a la especificación la siguiente cláusula:


```

Pre :  $min\_suma(l) \leq suma \leq max\_suma(l)$ 
fun  $min\_suma(l) : \mathbb{Z} = \sum_{i=0}^{|l|-1} \text{if } l[i] < 0 \text{ then } l[i] \text{ else } 0$  fi
fun  $max\_suma(l) : \mathbb{Z} = \sum_{i=0}^{|l|-1} \text{if } l[i] > 0 \text{ then } l[i] \text{ else } 0$  fi
      
```

 ¿Ahora es una especificación válida? Si no lo es, justificarlo con un ejemplo como en el punto anterior.
- Dar una precondición que haga correcta la especificación

Respuesta

- $l = \langle 9, 9, 9 \rangle$, $suma = 1$, si l contiene a $result$, entonces necesariamente va a sumar por lo menos 9, por lo que no puede valer 1 su suma.
- $l = \langle 9, 9, 9 \rangle$, $suma = 1$, si l contiene a $result$, entonces necesariamente va a sumar por lo menos 9, por lo que no puede valer 1 su suma, y además suma cumple la desigualdad $0 \leq suma \leq 27$
- proc** elementosQueSumen (in $l : seq\langle\mathbb{Z}\rangle$, in suma: \mathbb{Z} , out result : $seq\langle\mathbb{Z}\rangle$) {
 Pre { $cantSubSeqCumplenSuma(l, suma) > 0$ }
 Post {
 /* La secuencia result está incluida en la secuencia l */
 $(\forall x : \mathbb{Z})(x \in result \rightarrow \#apariciones(x, result) \leq \#apariciones(x, l))$
 /* La suma de la result coincide con el valor de la suma */
 $\wedge suma = \sum_{i=0}^{|result|-1} result[i]$
 }

$aux\ cantSubSeqCumplenSuma(l : seq\langle\mathbb{Z}\rangle, suma : \mathbb{Z}) : \mathbb{Z} =$
 $\sum_{j=1}^{|l|} \sum_{i=0}^{|l|-1} \text{if } (|subseq(l, i, j)| > 0 \wedge sumaSeq(subseq(l, i, j)) = suma) \text{ then } 1 \text{ else } 0$ fi
 $aux\ sumaSeq(l : seq\langle\mathbb{Z}\rangle) : \mathbb{Z} = \sum_{k=0}^{|l|-1} l[k]$

Ejercicio 3. ★ Para los siguientes problemas, dar todas las soluciones posibles a las entradas dadas.

- proc** raizCuadrada (in x: \mathbb{R} , out result: \mathbb{R}) {
 Pre { $x \geq 0$ }
 Post { $result^2 = x$ }
 }
 - $x = 0$
 - $x = 1$
 - $x = 27$
- ★
 proc indiceDelMaximo (in l: $seq\langle\mathbb{R}\rangle$, out result: \mathbb{Z}) {
 Pre { $|l| > 0$ }
 Post {
 $0 \leq result < |l|$
 $\wedge_L((\forall i : \mathbb{Z})(0 \leq i < |l| \rightarrow_L l[i] \leq l[result]))$
 }
 }

- I) $l = \langle 1, 2, 3, 4 \rangle$
- II) $l = \langle 15, 5, -18, 4, 215, 15, 5, -1 \rangle$
- III) $l = \langle 0, 0, 0, 0, 0, 0 \rangle$

c) ★

```

proc indiceDelPrimerMaximo (in l: seq( $\mathbb{R}$ ),out result:  $\mathbb{Z}$ ) {
  Pre  $\{|l| > 0\}$ 
  Post {
     $0 \leq result < |l|$ 
     $\wedge ((\forall i : \mathbb{Z})(0 \leq i < |l| \rightarrow_L (l[i] < l[result] \vee (l[i] = l[result] \wedge i \geq result))))$ 
  }
}

```

- I) $l = \langle 1, 2, 3, 4 \rangle$
- II) $l = \langle 15, 5, -18, 4, 215, 15, 5, -1 \rangle$
- III) $l = \langle 0, 0, 0, 0, 0, 0 \rangle$

d) ¿Para qué valores de entrada **indiceDelPrimerMaximo** y **indiceDelMaximo** tienen necesariamente la misma salida?

Respuesta

- a) I) $result = 0$
 II) $result = 1; -1$
 III) $result = 3\sqrt{3}; -3\sqrt{3}$
- b) I) $result = 3$
 II) Cualquier cosa, no dice nada cuando hay más de una aparición del maximo.
 III) Idem
- c) I) $result = 3$
 II) $result = 0$,
 III) $result = 0$
- d) Van a tener la misma salida cuando no haya más de una aparición del maximo en la lista (ya que en caso contrario **indiceDelMaximo** crashearia).

Ejercicio 4. ★ Sea $f : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ definida como:

$$f(a, b) = \begin{cases} 2b & \text{si } a < 0 \\ b - 1 & \text{en otro caso} \end{cases}$$

¿Cuáles de las siguientes especificaciones son correctas para el problema de calcular $f(x, y)$?
 Para las que no lo son, indicar por qué.

- a) **proc f** (in a, b: \mathbb{R} ,out result: \mathbb{R}) {
Pre $\{True\}$
Post {
 $(a < 0 \wedge result = 2 * b)$
 \wedge
 $(a \geq 0 \wedge result = b - 1)$
 }
 }
- b) **proc f** (in a, b: \mathbb{R} ,out result: \mathbb{R}) {
Pre $\{True\}$
Post $\{(a < 0 \wedge result = 2 * b) \vee (a > 0 \wedge result = b - 1)\}$
 }

- c) **proc f** (in a, b: \mathbb{R} , out result: \mathbb{R}) {
 Pre {*True*}
 Post $\{(a < 0 \wedge result = 2 * b) \vee (a \geq 0 \wedge result = b - 1)\}$
 }
- d) **proc f** (in a, b: \mathbb{R} , out result: \mathbb{R}) {
 Pre {*True*}
 Post {
 $(a < 0 \rightarrow result = 2 * b)$
 \wedge
 $(a \geq 0 \rightarrow result = b - 1)$
 }
 }
- e) **proc f** (in a, b: \mathbb{R} , out result: \mathbb{R}) {
 Pre {*True*}
 Post $\{(a < 0 \rightarrow result = 2 * b) \vee (a \geq 0 \rightarrow result = b - 1)\}$
 }
- f) **proc f** (in a, b: \mathbb{R} , out result: \mathbb{R}) {
 Pre {*True*}
 Post $\{result = (\text{if } a < 0 \text{ then } 2 * b \text{ else } b - 1 \text{ fi})\}$
 }

Respuesta

- a) Mal, por muchas razones que no tengo ganas de aclarar.
- b) Mal, tendria que ser $a \geq 0$ despues de la conjunción.
- c) Correcta
- d) Correcta
- e) Mmmmmmm.... creo que no, si alguna implicación falla no puedo devolver true.
- f) Correcta

Ejercicio 5. ★ Considerar la siguiente especificación, junto con un algoritmo que dado x devuelve x^2 .

proc unoMasGrande (in x: \mathbb{R} , out result: \mathbb{R}) {
 Pre {*True*}
 Post $\{result > x\}$
 }

- a) ¿Qué devuelve el algoritmo si recibe $x = 3$? ¿El resultado hace verdadera la postcondición de **unoMasGrande**?
- b) ¿Qué sucede para las entradas $x = 0,5$, $x = 1$, $x = 0,2$ y $x = -7$?
- c) Teniendo en cuenta lo respondido en los puntos anteriores, escribir una precondition para **unoMasGrande**, de manera tal que el algoritmo sea una implementación correcta.

Respuesta

- a) Segun lo que interpreto, el algoritmo esta tratando de cumplir la **Post**, entonces, al pasarle 3 devuelve un 9 que efectivamente cumple la **Post** ya que $9 > 3$.
- b)
- $x = 0,5; result = 0,25$, no cumple la **Post**.
 - $x = 1; result = 1$, no cumple la **Post**.
 - $x = 0,2; result = 0,04$, no cumple la **Post**.
 - $x = -7; result = 49$, cumple la **Post**.
- c) **Pre** $\{abs(x) > 1\}$

Ejercicio 6. ★ Sean x y r variables de tipo \mathbb{R} . Considerar los siguientes predicados:

$$\begin{array}{ll} P1 : \{x \leq 0\} & Q1 : \{r \geq x^2\} \\ P2 : \{x \leq 10\} & Q2 : \{r \geq 0\} \\ P3 : \{x \leq -10\} & Q3 : \{r = x^2\} \end{array}$$

- a) Indicar la relación de fuerza entre P1, P2 y P3.
- b) Indicar la relación de fuerza entre Q1, Q2 y Q3.
- c) Sea E1 la siguiente especificación. Escribir 2 programas que cumplan con E1.

```
proc hagoAlgo (in x:  $\mathbb{R}$ , out r:  $\mathbb{R}$ ) {
  Pre  $\{x \leq 0\}$ 
  Post  $\{r \geq x^2\}$ 
}
```

- d) Sea A un algoritmo que cumple con E1. Decidir si necesariamente cumple las siguientes especificaciones:

- a) **Pre:** $\{x \leq -10\}$, **Post:** $\{r \geq x^2\}$
- b) **Pre:** $\{x \leq 10\}$, **Post:** $\{r \geq x^2\}$
- c) **Pre:** $\{x \leq 0\}$, **Post:** $\{r \geq 0\}$
- d) **Pre:** $\{x \leq 0\}$, **Post:** $\{r = x^2\}$
- e) **Pre:** $\{x \leq -10\}$, **Post:** $\{r \geq 0\}$
- f) **Pre:** $\{x \leq 10\}$, **Post:** $\{r \geq 0\}$
- g) **Pre:** $\{x \leq -10\}$, **Post:** $\{r = x^2\}$
- h) **Pre:** $\{x \leq 10\}$, **Post:** $\{r = x^2\}$

- e) ¿Qué conclusión pueden sacar? ¿Qué debe cumplirse con respecto a las precondiciones y postcondiciones para que sea seguro reemplazar la especificación?

Respuesta

- a)
 - $P1 \rightarrow P2$ es contingencia.
 - $P1 \rightarrow P3$ es tautología.
 - $P2 \rightarrow P1$ es tautología.
 - $P2 \rightarrow P3$ es tautología.
 - $P3 \rightarrow P1$ es contingencia.
 - $P3 \rightarrow P2$ es contingencia.
- b)
 - $Q1 \rightarrow Q2$ es tautología.
 - $Q1 \rightarrow Q3$ es contingencia.
 - $Q2 \rightarrow Q1$ es contingencia.
 - $Q2 \rightarrow Q3$ es contingencia.
 - $Q3 \rightarrow Q1$ es tautología.
 - $Q3 \rightarrow Q2$ es tautología.
- c)
 - 1) Programa en lenguaje de especificación:

$$\text{aux programal}(x : \mathbb{R}) \text{res} : \mathbb{R} = x * x + 3$$
 - 2) Programa en Perl:


```
# !/usr/bin/perl
use v5.26;
my $x;
my $res;
chomp($x=< STDIN >);
if($x <= 0){
  $res=x * x + 1;
}
say $res;
```

- d) a) Cumple.
- b) No cumple.
- c) Cumple.
- d) No Cumple.
- e) Cumple.
- f) No cumple.
- g) No cumple.
- h) No cumple.

e) La nueva **Pre** Tiene que estar incluido en el rango de la **Pre** original,y ademas tiene que pasar lo mismo con las **Post**.

Ejercicio 7. ★ Considerar las siguientes dos especificaciones, junto con un algoritmo a que satisface la especificación de **p2**.

```
proc p1 (in x: ℝ,in n: ℤout result: ℤ) {
  Pre {x ≠ 0}
  Post {x^n - 1 < result ≤ x^n}
}
```

```
proc p2 (in x: ℝ,in n: ℤout result: ℤ) {
  Pre {n ≤ 0 → x ≠ 0}
  Post {result = ⌊x^n⌋}
}
```

- a) Dados valores de x y n que hacen verdadera la precondition de **p1**, demostrar que hacen también verdadera la precondition de **p2**.
- b) Ahora, dados estos valores de x y n , supongamos que se ejecuta a : llegamos a un valor de res que hace verdadera la postcondición de **p2**. ¿Será también verdadera la postcondición de **p1**?
- c) ¿Podemos concluir que a satisface la especificación de **p1**?

Respuesta

- a)
- b)
- c)

Ejercicio 8. Considerar las siguientes especificaciones:

```
proc n-esimo1 (in l: seq(ℝ),in n: ℤ,out result: ℤ) {
  Pre {
    /*Los elementos están ordenados */
  }
  Post {result^2 = x}
}
```

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 9. ★ Especificar los siguientes problemas:

- a) Dado un número entero, decidir si es par.
- b) Dado un entero n y uno m , decidir si n es un múltiplo de m .
- c) Dado un número real, devolver su inverso multiplicativo.
- d) Dada una secuencia de caracteres, obtener de ella sólo los que son numéricos (con todas sus apariciones sin umportar el orden de aparición).
- e) Dada una secuencia de reales, devolver la secuencia que resulta de duplicar sus valores en las posiciones impares.
- f) Dado un número entero, listar todos sus divisores positivos (sin duplicados).

Respuesta

- a)
- b)
- c)
- d)
- e) **proc DuplicaValoresEnImpares** (in $s : seq(\mathbb{R})$, out $m : seq(\mathbb{R})$) {
 Pre $\{|s| > 0\}$
 Post {
 $|s| = |m| \wedge_L$
 $(\forall i : \mathbb{Z})(0 \leq i < |s| \wedge_L i \bmod 2 = 1) \rightarrow_L (m[i] = s[i] * 2)$
 $\wedge (\forall j : \mathbb{Z})(0 \leq j < |s| \wedge_L j \bmod 2 = 0) \rightarrow_L (m[j] = s[j])$
 }
}
- f)

Ejercicio 10. Considerar el problema de decidir, dados n y m enteros, si n es múltiplo de m , y la siguiente especificación.

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 11. Considerar el problem de, dada una secuencia de números reales, devolver la que resulta de duplicar sus valores en las posiciones impares.

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 12. ★ Especificar el problema de dado un entero positivo retornar una secuencia de 0s y 1s que represente es número en base 2 (es decir, en binario).

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 13. Con lo visto en los ejercicios 9 a 12 ¿Encuentra casos de sub y sobreespecificación en las especificaciones del ejercicio 8?

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 14. Especificar los siguientes problemas:

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 15. Especificar los siguientes problemas sobre secuencias:

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 16. Especificar los siguientes problemas:

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Especificación de problemas usando inout

Ejercicio 17. ★ Dados dos enteros a y b , se necesita calcular su suma y retornarla en un entero c . ¿Cuáles de las siguientes especificaciones son correctas para este problema? Para las que no lo son, indicar por qué.

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 18. ★ Dada una secuencia l , se deas sacar su primer elemento y devolverlo. Decidir cuáles de estas especificaciones son correctas. Para las que no lo son, indicar por qué y justificar con ejemplos.

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 19. Considerar la siguiente especificación:

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 20. Explicar coloquialmente la siguiente especificación:

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 21. Dada una secuencia de enteros, se requiere multiplicar por 2 aquéllos valores que se encuentran en posiciones pares. Indicar por qué son incorrectas las siguientes especificaciones, y proponer una alternativa correcta

- a)
- b)
- c)
- d)

Respuesta

- a)
- b)
- c)
- d)

Ejercicio 22. Especificar los siguientes problemas de modificación de secuencias:

- a) ★ **proc primosHermanos**(inout $l : seq\langle \mathbb{Z} \rangle$), que dada una secuencia de enteros mayores a dos, reemplaza dichos valores por el número primo menor más cercano. Por ejemplo, si $l = \langle 6, 5, 9, 14 \rangle$, luego de aplicar **primosHermanos**(l), $l = \langle 5, 5, 7, 13 \rangle$
- b) ★ **proc reemplazar**(inout $l : \text{String}$, in $a, b : \text{Char}$), que reemplaza todas las apariciones de a en l por b .
- c) **proc recortar**(inout $l : seq\langle \mathbb{Z} \rangle$, in $a : \mathbb{Z}$), que saca de l todas las apariciones de a consecutivas que aparezcan al principio. Por ejemplo **recortar**($\langle 2, 2, 3, 2, 4 \rangle, 2$) = $\langle 3, 2, 4 \rangle$, mientras que **recortar**($\langle 2, 2, 3, 2, 4 \rangle, 3$) = $\langle 2, 2, 3, 2, 4 \rangle$.
- d) **proc intercambiarParesConImpares**(inout $l : \text{String}$), que toma una secuencia de longitud par y la modifica de modo tal que todas las posiciones de la forma $2k$ quedan intercambiadas con las posiciones $2k + 1$. Por ejemplo, **intercambiarParesConImpares**(“*adinle*”) modifica de la siguiente manera: “*daniel*”.
- e) ★ **proc limpiarDuplicados**(inout $l : seq\langle \text{Char} \rangle$, out $dup : seq\langle \text{Char} \rangle$), que elimina los elementos duplicados de l dejando sólo su primera aparición (en el orden original). Devuelve además, dup una secuencia con todas las apariciones eliminadas (en cualquier orden).

Respuesta

- a)
- b)
- c)
- d)

FIN.