



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(ℝ), in elem: ℝ, out result: ℤ) {
  Pre {elem ∈ 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(ℝ), out result: Bool) {
  Pre {True}
  Post {result = True ↔ ((∀i : ℤ)(0 ≤ i < |l| →L l[i] = 2 * l[i - 1]))}
}
```

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

```
proc minimo (in l: seq(ℝ), out result: ℤ) {
  Pre {True}
  Post {(∀y : ℤ)((y ∈ l ∧ y ≠ x) → 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(ℝ), in elem: ℝ, out result: ℤ) {
  Pre {elem ∈ l ∧ cantApariciones(elem, l) = 1}
  Post {0 ≤ result < |l| ∧L 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(ℝ), out result: Bool) {
  Pre {True}
  Post {result = True ↔ ((∀i : ℤ)(0 ≤ i < |l| - 1 →L 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(ℝ), out result: ℤ) {
  Pre {result ∈ l}
  Post {(∀y : ℤ)(y ∈ l → 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:

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

- Indicar la relación de fuerza entre P1, P2 y P3.
- Indicar la relación de fuerza entre Q1, Q2 y Q3.
- 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  $\{r \geq x^2\}$ 
}
```

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

- Pre:** $\{x \leq -10\}$, **Post:** $\{r \geq x^2\}$
- Pre:** $\{x \leq 10\}$, **Post:** $\{r \geq x^2\}$
- Pre:** $\{x \leq 0\}$, **Post:** $\{r \geq 0\}$
- Pre:** $\{x \leq 0\}$, **Post:** $\{r = x^2\}$
- Pre:** $\{x \leq -10\}$, **Post:** $\{r \geq 0\}$
- Pre:** $\{x \leq 10\}$, **Post:** $\{r \geq 0\}$
- Pre:** $\{x \leq -10\}$, **Post:** $\{r = x^2\}$
- Pre:** $\{x \leq 10\}$, **Post:** $\{r = x^2\}$

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

Respuesta

- $P1 \rightarrow P2$ 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.
- $Q1 \rightarrow Q2$ 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.
- Programa en lenguaje de especificación:
 $\text{aux programal}(x : \mathbb{R}) \text{res} : \mathbb{R} = x * x + 3$
 - 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: ℤ,out result: ℤ) {
  Pre {x ≠ 0}
  Post {x^n - 1 < result ≤ x^n}
}
```

```
proc p2 (in x: ℝ,in n: ℤ,out 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)

$$\begin{aligned}
 x \neq 0 \rightarrow (n \leq 0 \rightarrow x \neq 0) &= True \rightarrow (n \leq 0 \rightarrow True) \\
 &= True \rightarrow (True) \\
 &= True
 \end{aligned}$$

b) Sep.

- c) No, el item anterior valia, porque partiamos del supuesto que las variables cumplan la pre de **p1**, sin esa restricción entonces ya no es verdad que a satisface la especificación de **p1**.
 $n = 4$ y $x = 0$ cumple **p2**, pero no cumple **p1**.

Ejercicio 8. Considerar las siguientes especificaciones:

```
proc n-esimo1 (in l: seq(ℝ),in n: ℤ,out result: ℤ) {
  Pre {
    /*Los elementos están ordenados */
    (∀i: ℤ)(0 ≤ i < |l| - 1 → l[i] < l[i + 1])
    ∧ 0 ≤ n < |l|
  }
  Post {result = l[n]}
}
```

```
proc n-esimo2 (in l: seq(ℝ),in n: ℤ,out result: ℤ) {
  Pre {
    /*Los elementos son distintos entre si */
    (∀i: ℤ)(0 ≤ i < |l| → (∀j: ℤ)(0 ≤ j < |l| ∧ i ≠ j → l[i] ≠ l[j]))
    ∧ 0 ≤ n < |l|
  }
```

```

}
Post {
  resultl ∈ l
  ∧
  n =  $\sum_{i=0}^{|l|-1}$  (if l[i] < result then 1 else 0 fi)
}
}

```

¿Es cierto que todo algoritmo que cumple con **n-esimo1** cumple también con **n-esimo2**? ¿Y al revés?
Sugerencia: Razonar de manera análoga a la del ejercicio anterior.

Respuesta

$l = \langle 1, 4, 6, 7 \rangle$, y $n = 3$ cumple **n-esimo1** y **n-esimo2**.
 Pero $l = \langle 4, 1, 6, 7 \rangle$, y $n = 3$ cumple **n-esimo2** y no **n-esimo1**.
 Detalle, encontrar un contraej, me garantiza que no es verdad que si cumple **n-esimo2** entonces va a cumplir **n-esimo1**, sin embargo no ocurre lo mismo para el otro caso. Lo dejo así, porque es intuible que es verdad.

Ejercicio 9. ★ Especificar los siguientes problemas:

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

Respuesta

- proc esPar** (in a: \mathbb{Z} , out res: **Bool**) {
 Pre {*True*}
 Post { $c = a \bmod 2$ }
 }
 - proc esMultiplo** (in n,m: \mathbb{Z} , out result: **Bool**) {
 Pre { $m \neq 0 \vee (m = 0 \wedge n = 0)$ }
 Post { $result = (n \bmod m = 0)$ }
 }
 - proc inversoMul**(in x: \mathbb{R} , out res: \mathbb{R}) {
 Pre { $x \neq 0$ }
 Post { $res = x^{-1}$ }
 }
 - proc obtenerCaracteres** (in l: $seq\langle \mathbf{Char} \rangle$ out res: $seq\langle \mathbf{Char} \rangle$) {
 Pre { $|l| > 0$ }
 Post {
 $(\forall i : \mathbb{Z})(0 \leq i < |l| \wedge_L esDigito(l[i]))$
 $\rightarrow_L cantAparicionesDeXEnl(l[i], l) = cantAparicionesDeXEnl(l[i], res)$
 }
 }
- aux cantAparicionesDeXEnl** (c: **Char** ,l: $seq\langle \mathbf{Char} \rangle$): \mathbb{Z})
 $= \sum_{i=0}^{|l|-1} (\text{if } l[i] = c \text{ then } 1 \text{ else } 0 \text{ fi})$

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) **proc divisoresPositivos** (in $a : \mathbb{Z}$, out $res : seq(\mathbb{Z})$) {
 Pre $\{a \neq 0\}$
 Post $\{(\forall i : \mathbb{Z})((1 \leq i \leq a \wedge_L a \bmod i = 0) \rightarrow_L i \in res) \wedge_L sinRepetidos(res)\}$
 }

pred sinRepetidos: $seq(\mathbb{Z})$ {
 $(\forall i, j : \mathbb{Z})(0 \leq i, j < |l| \rightarrow_L l[i] \neq l[j])$ }

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

proc esMultiplo (in $n, m : \mathbb{Z}$, out $result : \mathbf{Bool}$) {
 Pre $\{m \neq 0\}$
 Post $\{result = (n \bmod m = 0)\}$
 }

- Segun la definición matemática de múltiplo, ¿tiene sentido preguntarse si 4 es múltiplo de 0? ¿Cuál es la respuesta?
- ¿Debería ser $n = 4, m = 0$ una entrada válida para el problema? ¿Lo es en esta especificación?
- Corregir la especificación de manera tal que $n = 4, m = 0$ satisfaga la precondition (¡cuidado con las indefiniciones!).
- ¿Qué relación de fuerza hay entre la precondition nueva y la original?

Respuesta

- Yo que se, ya no me acuerdo si lo dieron en Algebra I, segun Wolfram Alpha da indefinido.
- No, si $m = 0$ entonces para cualquier n va a devolver indefinido, salvo para $n = 0$, ya que el 0 es el unico multiplo de 0.
- Pre** $\{m \neq 0 \vee (m = 0 \wedge n = 0)\}$
- La original implica la nueva.

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

- Para la secuencia $\langle 1, 2, 3, 4 \rangle$, ¿es $\langle 0, 4, 0, 8 \rangle$ un resultado correcto?
- Sea la siguiente especificación:

proc duplicarEnImpares (in $l : seq(\mathbb{R})$, out $result : seq(\mathbb{R})$) {
 Pre $\{True\}$
 Post $\{|result| = |l| \wedge (\forall i : \mathbb{Z})(0 \leq i < |result| \wedge i \bmod 2 = 1) \rightarrow_L result[i] = 2 * l[i]\}$
 }

Si $l = \langle 1, 2, 3, 4 \rangle$, ¿ $result = \langle 0, 4, 0, 8 \rangle$ satisface la postcondición?

- Si es necesario, corregir la especificación para que describa correctamente el resultado esperado.
- ¿Qué relación de fuerza hay entre la nueva postcondición y la original?

Respuesta

-
-
-
-

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).

Respuesta

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

Respuesta

Ejercicio 14. Especificar los siguientes problemas:

- a) ★ Dado un número entero positivo, obtener la suma de sus factores primos.
- b) Dado un número entero positivo, decidir si es perfecto. Se dice que un número es perfecto cuando es igual a la suma de sus divisores (excluyéndose a sí mismo).
- c) Dado un número entero positivo n , obtener el menor entero positivo $m > 1$ tal que m sea coprimo con n .
- d) ★ Dado un entero positivo, obtener su descomposición en factores primos. Devolver una secuencia de tuplas (p, c) , donde p es un factor primo y c es su exponente, ordenada en forma creciente con respecto a p .
- e) Dada una secuencia de números reales, obtener la diferencia máxima entre dos de sus elementos.
- f) ★ Dada una secuencia de números enteros, devolver aquel que divida a más elementos de dicha secuencia. El elemento tiene que pertenecer a la secuencia original. Si existe más de un elemento que cumple esta propiedad, devolver alguno de ellos.

Respuesta

- a)
- b)
- c)
- d)
- e)
- f)

Ejercicio 15. Especificar los siguientes problemas sobre secuencias:

- a) **proc nEsimaAparicion**(in $l : seq(\mathbb{R})$, in $e : \mathbb{R}$, in $n : \mathbb{Z}$, out $result : \mathbb{Z}$), que devuelve el índice de la n -ésima aparición de e en l .
- b) Dadas dos secuencias s y t , decidir si s es una subcadena de t .
- c) ★ Dadas dos secuencias s y t , decidir si s está *incluida* en t , es decir, si todos los elementos de s aparecen en t en igual o mayor cantidad.
- d) **proc mezclarOrdenado**(in $s, t : seq(\mathbb{Z})$, out $result : seq(\mathbb{Z})$) que recibe dos secuencias ordenadas y devuelve el resultado de intercalar sus elementos de manera ordenada.
- e) Dadas dos secuencias s y t especificar el procedimiento *intersecciónSinRepetidos* que retorna una secuencia que contiene únicamente los elementos que aparecen en ambas secuencias.
- f) ★ Dadas dos secuencias s y t , devolver su *intersección*, es decir, una secuencia con todos los elemntos que aparecen en ambas. Si un mismo elemento tiene repetidos, la secuencia retornada debe contener la cantidad mínima de apariciones en de s y de t .

Respuesta

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

Ejercicio 16. Especificar los siguientes problemas:

- a) **proc cantApariciones**(in $l : \text{String}$, out $result : seq(\text{Char} \times \mathbb{Z})$) que devuelve la secuencia con todos los elementos de l , sin duplicados con su cantidad de apariciones (en un orden cualquiera). Ejemplos:
- $cantApariciones (\langle 'a' \rangle) = \langle \langle 'a', 1 \rangle \rangle$
 - $cantApariciones (\langle 'a', 'b', 'c' \rangle) = \langle \langle 'a', 1 \rangle, \langle 'c', 1 \rangle, \langle 'b', 1 \rangle \rangle$
 - $cantApariciones (\langle 'a', 'b', 'c', 'b', 'd', 'b' \rangle) = \langle \langle 'a', 1 \rangle, \langle 'b', 3 \rangle, \langle 'd', 1 \rangle, \langle 'c', 1 \rangle \rangle$
 - $cantApariciones (\langle \rangle) = \langle \rangle$
- b) Dada una secuencia, devolver una secuencia de secuencias que contenga todos sus prefijos, en ordn creciente de longitud.
- c) ★ Dada una secuencia de secuencias de enteros l , devolver una secuencia de l que contenga el máximo valor. Por ejemplo, si $l = \langle \langle 2, 3, 5 \rangle, \langle 8, 1 \rangle, \langle 2, 8, 4, 3 \rangle \rangle$, devolver $\langle 8, 1 \rangle$ o $\langle 2, 8, 4, 3 \rangle$.
- d) **proc interseccionMultiple**(in $ls : seq(seq(\mathbb{Z}))$, out $l : seq(\langle \mathbb{R} \rangle)$) que devuelve en l el resultado de la intersección de todas las secuencias de ls .
- e) ★ Dada una secuencia l con todos sus elementos distintos, devolver la secuencia de *partes*, es decir, la secuencia de todas las secuencias incluidas en l , cada una con sus elementos en el mismo orden en que aparecen en l .

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) **proc sumar** (inout $a, b, c : \mathbb{Z}$) {
 Pre $\{True\}$
 Post $\{a + b = c\}$
 }
- b) **proc sumar** (in $a, b : \mathbb{Z}$, in $c : \mathbb{Z}$) {
 Pre $\{True\}$
 Post $\{c = a + b\}$
 }
- c) **proc sumar** (in $a, b : \mathbb{Z}$, out $c : \mathbb{Z}$) {
 Pre $\{True\}$
 Post $\{c = a + b\}$
 }
- d) **proc sumar** (inout $a, b : \mathbb{Z}$, out $c : \mathbb{Z}$) {
 Pre $\{a = A_0 \wedge b = B_0\}$
 Post $\{a = A_0 \wedge b = B_0 \wedge c = a + b\}$
 }

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) **proc tomarPrimero** (inout l : $seq(\mathbb{R})$,out result: \mathbb{R}) {
 Pre $\{|l| > 0\}$
 Post $\{result = head(l)\}$
 }
- b) **proc tomarPrimero** (inout l : $seq(\mathbb{R})$,out result: \mathbb{R}) {
 Pre $\{|l| > 0 \wedge l = L_0\}$
 Post $\{result = head(L_0)\}$
 }
- c) **proc tomarPrimero** (inout l : $seq(\mathbb{R})$,out result: \mathbb{R}) {
 Pre $\{|l| > 0\}$
 Post $\{result = head(L_0) \wedge |l| = |L_0| - 1\}$
 }
- d) **proc tomarPrimero** (inout l : $seq(\mathbb{R})$,out result: \mathbb{R}) {
 Pre $\{|l| > 0 \wedge l = L_0\}$
 Post $\{result = head(L_0) \wedge l = tail(L_0)\}$
 }
- e) **proc tomarPrimero** (inout l : $seq(\mathbb{R})$,out result: \mathbb{R}) {
 Pre $\{|l| > 0 \wedge l = L_0\}$
 Post {
 $result = head(L_0)$
 $\wedge |l| = |L_0| - 1$
 $\wedge_L (\forall i : \mathbb{Z})(0 \leq i < |l| \rightarrow_L l[i] = L_0[i + 1])$
 }
 }

Respuesta

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

Ejercicio 19. Considerar la siguiente especificación:

```
proc intercambiar (inout  $l$ :  $seq(\mathbb{R})$ ,in  $i, j : \mathbb{Z}$ ) {
  Pre  $\{0 \leq i < |l| \wedge 0 \leq j < |l| \wedge l = L_0\}$ 
  Post {
    /*Las secuencias tienen la misma longitud*/
     $|l| = |L_0|$ 
     $\wedge$ 
    /*Intercambia i*/
     $l[i] = L_0[j]$ 
     $\wedge$ 
    /*Intercambia j*/
     $l[j] = L_0[i]$ 
  }
}
```

- a) ¿Esta especificación es válida? Si lo es ¿qué problema describe?
- b) Mostrar con un ejemplo que la postcondicion está sub-especificada (es decir, que hay valores que la hacen verdadera aunque no son deseables como solución).
- c) Corregir la especificación agregando a la postcondición una o más cláusulas **Post:** .

Respuesta

- a)
- b)
- c)

Ejercicio 20. Explicar coloquialmente la siguiente especificación:

```
proc copiarPrimero (inout l: seq⟨ℝ⟩, inout i: ℤ) {  
  Pre {  
    /*Valores iniciales*/  
    l = L0 ∧ i = I0  
    ∧  
    /*Secuencia no vacia*/  
    |l| > 0  
    ∧  
    /*Indice en rango*/  
    0 ≤ i < |l|  
  }  
  Post {  
    l[I0] = L0[0]  
    ∧  
    i = L0[I0]  
    ∧  
    ((∀j : ℤ)(0 ≤ j < |l| ∧ j ≠ I0) →L l[j] = L0[I0])  
  }  
}
```

Respuesta

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) **proc** duplicarPares (inout l: seq⟨ℤ⟩) {
 Pre {l = L₀}
 Post {
 |l| = |L₀|
 ∧
 (∀i : ℤ)(0 ≤ i < |l| ∧ i mod 2 = 0) →_L l[i] = 2 * L₀[i]
 }
}
- b) **proc** duplicarPares (inout l: seq⟨ℤ⟩) {
 Pre {l = L₀}
 Post {(∀i : ℤ)(0 ≤ i < |l| ∧ i mod 2 ≠ 0) →_L l[i] = L₀[i]
 ∧
 (∀i : ℤ)(0 ≤ i < |l| ∧ i mod 2 = 0) →_L l[i] = 2 * L₀[i]
 }
}
- c) **proc** duplicarPares (inout l: seq⟨ℤ⟩, out result: seq⟨ℤ⟩) {
 Pre {True}
 Post {|l| = |result|
 ∧
 (∀i : ℤ)(0 ≤ i < |l| ∧ i mod 2 ≠ 0) →_L result[i] = l[i]
 ∧
 (∀i : ℤ)(0 ≤ i < |l| ∧ i mod 2 = 0) →_L result[i] = 2 * l[i]
 }
}

Respuesta

- a)
- b)
- c)

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.