



Guía Práctica 3 Especificación de problemas

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 de ésta, devuelve en *result* alguna posición de la secuencia en la cual se encuentre 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⟨Z⟩, out result: Bool) {  
    Pre {True}  
    Post {result = True ↔ ((∀i : Z)(0 ≤ i < |l| → l[i] = 2 * l[i - 1]))}  
}
```

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

```
proc minimo (in l: seq⟨Z⟩, out result: Z) {  
    Pre {True}  
    Post {(∀y : Z)((y ∈ l ∧ y ≠ x) → y > result)}  
}
```

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

```
proc elementosQueSumen (in l: seq⟨Z⟩, in suma: Z, out result: seq⟨Z⟩) {  
    Pre {True}  
    Post {  
        /* La secuencia result está incluida en la secuencia l */  
        (∀x : Z)(x ∈ result → #apariciones(x, result) ≤ #apariciones(x, l))  
        /* La suma de la result coincide con el valor suma */  
        ∧ suma = ∑i=0|result|-1 result[i]  
    }  
}
```

- a) Mostrar valores para *l* y *suma* que hagan verdadera la precondition, pero tales que no exista *result* que cumpla la postcondición.

- b) Supongamos que agregamos a la especificación la siguiente cláusula:

```
Pre : min_suma(l) ≤ suma ≤ max_suma(l)  
fun min_suma(l) : Z = ∑i=0|l|-1 if l[i] < 0 then l[i] else 0 fi  
fun max_suma(l) : Z = ∑i=0|l|-1 if l[i] > 0 then l[i] else 0 fi
```

¿Ahora es una especificación válida? Si no lo es, justificarlo con un ejemplo como en el punto anterior.

c) Dar una precondition que haga correcta la especificación.

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

a) `proc raizCuadrada` (in $x:\mathbb{R}$, out $result:\mathbb{R}$) {
 `Pre` $\{x \geq 0\}$
 `Post` $\{result^2 = x\}$
}

I) $x = 0$

II) $x = 1$

III) $x = 27$

b) ★

`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` ($l: seq\langle\mathbb{R}\rangle$, $result:\mathbb{Z}$) {
 `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?

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* = (if $a < 0$ then $2 * b$ else $b - 1$ fi)}
}

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.

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

P1: $\{x \leq 0\}$
 P2: $\{x \leq 10\}$
 P3: $\{x \leq -10\}$

Q1: $\{r \geq x^2\}$
 Q2: $\{r \geq 0\}$
 Q3: $\{r = x^2\}$

- 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: R, out r: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?

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  $\{x \neq 0\}$ 
  Post  $\{x^n - 1 < result \leq x^n\}$ 
}
```

```
proc p2 (in x: R, in n: Z, out result:Z) {
  Pre  $\{n \leq 0 \rightarrow x \neq 0\}$ 
  Post  $\{result = \lfloor x^n \rfloor\}$ 
}
```

- Dados valores de x y n que hacen verdadera la precondición de p1, demostrar que hacen también verdadera la precondición de p2.
- 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?
- ¿Podemos concluir que a satisface la especificación de p1?

Ejercicio 8. Considerar las siguientes especificaciones:

```
proc n-esimo1 (in l: seq(Z), in n: Z, out result: Z) {
  Pre {
    /*Los elementos están ordenados*/
     $(\forall i: Z)(0 \leq i < |l| - 1 \rightarrow_L l[i] < l[i + 1])$ 
     $\wedge 0 \leq n < |l|$ 
  }
}
```

```

    Post {result = l[n]}
}

proc n-esimo2 (in l: seq(Z), in n: Z, out result:Z) {
    Pre {
        /*Los elementos son distintos entre sí*/
        (∀i : Z)(0 ≤ i < |l| →L ((∀j : Z)(0 ≤ j < |l| ∧ i ≠ j) →L l[i] ≠ l[j]))
        ∧
        0 ≤ n < |l|
    }
    Post {
        result ∈ l
        ∧
        n = ∑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.

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).

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: Z, out result:Bool) {
    Pre {m ≠ 0}
    Post {result = (n mód m = 0)}
}

```

- Según 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?

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(R), out result: seq(R)) {
    Pre {True}
    Post {|result| = |l| ∧ (∀i : Z)((0 ≤ i < |result| ∧ i mód 2 = 1) →L result[i] = 2 * l[i])}
}

```

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

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

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

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

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, e) , donde p es un factor primo y e 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.

Ejercicio 15. Especificar los siguientes problemas sobre secuencias:

- a) `proc nEsimaAparicion(in l : seq⟨R⟩, in e : R, in n : Z, out result : 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⟨Z⟩, out result : seq⟨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 elementos que aparecen en ambas. Si un mismo elemento tiene repetidos, la secuencia retornada debe contener la cantidad mínima de apariciones en s y de t .

Ejercicio 16. Especificar los siguientes problemas:

- a) `proc cantApariciones(in l : seq⟨Char⟩, out result : seq⟨Char × Z⟩)` que devuelve la secuencia con todos los elementos de l , sin duplicados, con su cantidad de apariciones (en un orden cualquiera). Ejemplos:
 - $\text{cantApariciones}(\langle 'a' \rangle) = \langle \langle 'a', 1 \rangle \rangle$
 - $\text{cantApariciones}(\langle 'a', 'b', 'c' \rangle) = \langle \langle 'a', 1 \rangle, \langle 'c', 1 \rangle, \langle 'b', 1 \rangle \rangle$
 - $\text{cantApariciones}(\langle 'a', 'b', 'c', 'b', 'd', 'b' \rangle) = \langle \langle 'a', 1 \rangle, \langle 'b', 3 \rangle, \langle 'd', 1 \rangle, \langle 'c', 1 \rangle \rangle$
 - $\text{cantApariciones}(\langle \rangle) = \langle \rangle$
- b) Dada una secuencia, devolver una secuencia de secuencias que contenga todos sus prefijos, en orden 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⟨R⟩⟩, out l : seq⟨R⟩)` 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 .

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 { $a = A_0 \wedge b = B_0 \wedge c = a + b$ }`
}

Ejercicio 18. ★ Dada una secuencia l , se desea 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: $seq\langle\mathbb{R}\rangle$, out result: \mathbb{R}) {`
 `Pre { $|l| > 0$ }`
 `Post { $result = head(l)$ }`
}
- b) `proc tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$, out result: \mathbb{R}) {`
 `Pre { $|l| > 0 \wedge l = L_0$ }`
 `Post { $result = head(L_0)$ }`
}
- c) `proc tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$, out result: \mathbb{R}) {`
 `Pre { $|l| > 0$ }`
 `Post { $result = head(L_0) \wedge |l| = |L_0| - 1$ }`
}
- d) `proc tomarPrimero (inout l: $seq\langle\mathbb{R}\rangle$, 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(R), out result:R) {
    Pre { |l| > 0 ∧ l = L0 }
    Post {
        result = head(L0)
        ∧ |l| = |L0| - 1
        ∧L ((∀ i : Z)(0 ≤ i < |l| →L l[i] = L0[i + 1]))
    }
}
```

Ejercicio 19. Considerar la siguiente especificación:

```
proc intercambiar (inout l: seq(R), in i, j: Z) {
    Pre { 0 ≤ i < |l| ∧ 0 ≤ j < |l| ∧ l = L0 }
    Post {
        /*Las secuencias tienen la misma longitud*/
        |l| = |L0|
        ∧
        /*Intercambia i*/
        l[i] = L0[j]
        ∧
        /*Intercambia j*/
        l[j] = L0[i]
    }
}
```

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

Ejercicio 20. Explicar coloquialmente la siguiente especificación:

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


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|`
 \wedge
 $(\forall i : \mathbb{Z})(0 \leq i < |l| \wedge i \bmod 2 = 0) \rightarrow_L l[i] = 2 * L_0[i]$
 `}`
`}`
- b) `proc duplicarPares (inout l: seq(Z)) {`
 `Pre {l = L0}`
 `Post {(\forall i : \mathbb{Z})(0 \leq i < |l| \wedge i \bmod 2 \neq 0) \rightarrow_L l[i] = L_0[i])`
 \wedge
 $(\forall i : \mathbb{Z})(0 \leq i < |l| \wedge i \bmod 2 = 0) \rightarrow_L l[i] = 2 * L_0[i]$
 `}`
`}`
- c) `proc duplicarPares (inout l: seq(Z), out result:seq(Z)) {`
 `Pre {True}`
 `Post {|l| = |result|`
 \wedge
 $(\forall i : \mathbb{Z})(0 \leq i < |l| \wedge i \bmod 2 \neq 0) \rightarrow_L result[i] = l[i]$
 \wedge
 $(\forall i : \mathbb{Z})(0 \leq i < |l| \wedge i \bmod 2 = 0) \rightarrow_L result[i] = 2 * l[i]$
 `}`
`}`

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

- a) ★ `proc primosHermanos(inout l : seq(Z))`, 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)`, $l = \langle 5, 3, 7, 13 \rangle$
- b) ★ `proc reemplazar(inout l : seq(Char), in a, b : Char)`, que reemplaza todas las apariciones de a en l por b .
- c) `proc recortar(inout l : seq(Z), in a : 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 : seq(Char))`, 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(Char), out dup : seq(Char))`, 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).