



DEPARTAMENTO  
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

## Práctica 3

2do cuatrimestre 2021 (virtual)

Algoritmos y Estructuras de Datos 1

Integrante	LU	Correo electrónico
Jonathan Bekenstein	348/11	<a href="mailto:jbekenstein@dc.uba.ar">jbekenstein@dc.uba.ar</a>



**Facultad de Ciencias Exactas y Naturales**

Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

<http://www.exactas.uba.ar>

# Índice

<b>3. Práctica 3</b>	<b>2</b>
3.1. Ejercicio 1 . . . . .	2
3.2. Ejercicio 2 . . . . .	2
3.3. Ejercicio 3 . . . . .	3
3.4. Ejercicio 4 . . . . .	3
3.5. Ejercicio 5 . . . . .	4
3.6. Ejercicio 6 . . . . .	4
3.7. Ejercicio 7 . . . . .	5
3.8. Ejercicio 8 . . . . .	5
3.9. Ejercicio 9 . . . . .	6
3.10. Ejercicio 10 . . . . .	7

### 3. Práctica 3

#### 3.1. Ejercicio 1

##### 3.1.A. Pregunta A

El problema es que la postcondición se puede indefinir si *result* está fuera del rango de la secuencia. Y eso no puede suceder nunca, las pre y post condiciones solo pueden ser verdaderas o falsas, nunca indefinidas.

```
proc buscar (in l: seq⟨ℝ⟩, in elem: ℝ, out result: ℤ) {  
  Pre {elem ∈ l}  
  Post {0 ≤ result < |l| ∧L l[result] = elem}  
}
```

##### 3.1.B. Pregunta B

El problema es que se indefinir al indexar  $l[i - 1]$  cuando  $i = 0$ . Como queremos verificar que el elemento en el índice  $i$  sea el doble que el elemento en el índice  $i - 1$ , tenemos que arrancar a revisar desde  $i = 1$ . Si la secuencia tiene un único elemento, entonces no hay que revisar nada pues el primer número de la progresión geométrica no va a ser el doble de nadie.

```
proc progresionGeometricaFactor2 (in l: seq⟨ℤ⟩, out result: Bool) {  
  Pre {True}  
  Post {result = True ↔ ((∀i : ℤ)(1 ≤ i < |l| →L l[i] = 2 * l[i - 1]))}  
}
```

##### 3.1.C. Pregunta C

El problema es que en la postcondición se pide  $y \neq x$  pero en el contexto de esta especificación,  $x$  no está definido. En cambio, lo que habría que pedir es que  $y \neq result$  o más simple aún, quitar esa condición y pedir  $y \geq result$ . A su vez, también falta especificar que  $result \in l$  para garantizar que  $result$  realmente sea un elemento de la secuencia.

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

#### 3.2. Ejercicio 2

##### 3.2.A. Pregunta A

Por ejemplo  $l = \langle 1 \rangle$ ,  $suma = 2$ . Cumplen la precondition que es simplemente *True* (o sea, cualquiera cosa cumple la precondition). Pero no existe forma de cumplir con la postcondición ya que no hay suficientes elementos en  $l$  para que sumados den 2.

##### 3.2.B. Pregunta B

Sigue siendo inválida porque solo restringe el valor máximo y mínimo que puede tener *suma* pero no garantiza que efectivamente existan elementos en  $l$  que sumados den *suma*. Por ejemplo  $l = \langle 1, 3 \rangle$ ,  $suma = 2$ . Con estos valores se cumple la precondition:  $\min\_suma(l) \leq suma \leq \max\_suma(l) \leftrightarrow 0 \leq 2 \leq 3$  pero no existen elementos en  $l$  que sumados den exactamente 2.

##### 3.2.C. Pregunta C

$(\exists s : seq\langle\mathbb{Z}\rangle)((\forall x : \mathbb{Z})(\#apariciones(x, s) \leq \#apariciones(x, l)) \wedge suma = \sum_{i=0}^{|s|-1} s[i])$

### 3.3. Ejercicio 3

#### 3.3.A. Pregunta A

- I)  $x = 0 \rightarrow result \in \{0\}$
- II)  $x = 1 \rightarrow result \in \{-1, 1\}$
- III)  $x = 27 \rightarrow result \in \{-\sqrt{27}, \sqrt{27}\}$

#### 3.3.B. Pregunta B

- I)  $l = \langle 1, 2, 3, 4 \rangle \rightarrow result \in \{3\}$
- II)  $l = \langle 15.5, -18, 4.215, 15.5, -1 \rangle \rightarrow result \in \{0, 3\}$
- III)  $l = \langle 0, 0, 0, 0, 0, 0 \rangle \rightarrow result \in \{0, 1, 2, 3, 4, 5\}$

#### 3.3.C. Pregunta C

- I)  $l = \langle 1, 2, 3, 4 \rangle \rightarrow result = 3$
- II)  $l = \langle 15.5, -18, 4.215, 15.5, -1 \rangle \rightarrow result = 0$
- III)  $l = \langle 0, 0, 0, 0, 0, 0 \rangle \rightarrow result = 0$

#### 3.3.D. Pregunta D

*indiceDelPrimerMaximo* y *indiceDelMaximo* tienen necesariamente la misma salida cuando no hay valores repetidos en la secuencia  $l$ . En estos casos, sería cuando  $l = \langle 1, 2, 3, 4 \rangle$ .

### 3.4. Ejercicio 4

#### 3.4.A. Pregunta A

Incorrecta porque las 2 expresiones deberían estar unidas con un  $\vee$ , ya que sino es imposible que se cumplan ambas al mismo tiempo (pues piden  $a < 0$  y también  $a \geq 0$ ).

#### 3.4.B. Pregunta B

Incorrecta porque la postcondición no contempla el caso cuando  $a = 0$ .

#### 3.4.C. Pregunta C

Correcta.

#### 3.4.D. Pregunta D

Correcta.

#### 3.4.E. Pregunta E

Incorrecta porque cuando  $a \geq 0$ , la implicación  $a < 0 \rightarrow result = 2 * b$  resulta *True* pues no se cumple el antecedente. Y luego como las 2 implicaciones están unidas con un  $\vee$ , este *True* ya hace que toda la postcondición sea *True* sin importar si efectivamente  $result = b - 1$  como debería ser según la especificación. Pasa lo mismo de forma análoga cuando  $a < 0$ .

### 3.4.F. Pregunta F

Correcta.

## 3.5. Ejercicio 5

### 3.5.A. Pregunta A

Si recibe  $x = 3$  devuelve  $result = 9$ , lo cual hace verdadera la postcondición pues  $9 > 3$ .

### 3.5.B. Pregunta B

$$x = 0.5 \rightarrow result = 0.5^2 = 0.25 \not> 0.5$$

$$x = 1 \rightarrow result = 1^2 = 1 \not> 1$$

$$x = -0.2 \rightarrow result = (-0.2)^2 = 0.04 > -0.2$$

$$x = -7 \rightarrow result = (-7)^2 = 49 > -7$$

### 3.5.C. Pregunta C

```
proc unoMasGrande (in x: ℝ, out result: ℝ) {  
    Pre { $x < 0 \vee x > 1$ }  
    Post { $result > x$ }  
}
```

## 3.6. Ejercicio 6

### 3.6.A. Pregunta A

$$P3 > P1 > P2$$

### 3.6.B. Pregunta B

$$Q3 > Q1 > Q2$$

### 3.6.C. Pregunta C

Programa 1:  $r := x * x$

Programa 2:  $r := x * x + 1$

### 3.6.D. Pregunta D

- a) Cumple porque la nueva precondition (P3) es más fuerte que la precondition original (P1).
- b) No cumple porque la nueva precondition (P2) es más débil que la precondition original (P1).
- c) Cumple porque la nueva postcondition (Q2) es más débil que la postcondition original (Q1).
- d) No cumple porque la nueva postcondition (Q3) es más fuerte que la postcondition original (Q1).
- e) Cumple porque la nueva precondition (P3) es más fuerte que la precondition original (P1) y la nueva postcondition (Q2) es más débil que la postcondition original (Q1).
- f) No cumple porque la nueva precondition (P2) es más débil que la precondition original (P1).

- g) No cumple porque la nueva postcondición (Q3) es más fuerte que la postcondición original (Q1).
- h) No cumple porque la nueva precondición (P2) es más débil que la precondición original (P1) y además la nueva postcondición (Q3) es más fuerte que la postcondición original (Q1).

### 3.6.E. Pregunta E

Dado un algoritmo que cumple con una especificación, es posible reemplazar dicha especificación por otra y que el algoritmo siga cumpliendo si:

- 1) La nueva precondición es más fuerte que la original y la nueva postcondición es más débil que la original.
- 2) La nueva precondición es más fuerte que la original y la postcondición se mantiene igual
- 3) La precondición se mantiene igual y la nueva postcondición es más débil que la original.

## 3.7. Ejercicio 7

### 3.7.A. Pregunta A

Sabiendo que vale la precondición de  $p1$  se puede afirmar que  $x \neq 0$ .

Luego, se puede dividir en 2 casos para ver cuándo vale la precondición de  $p2$ :

- 1) Si  $n > 0$  el antecedente de la implicación es falso y así la implicación resulta verdadera, sin importar el valor de  $x$ .
- 2) Si  $n \leq 0$  la implicación resulta verdadera si  $x \neq 0$ . Esto vale pues sabemos que se cumple la precondición de  $p1$ .

Por lo tanto vale la precondición de  $p2$ .

*Nota: Me parece poco formal esta "demostración".*

### 3.7.B. Pregunta B

En esencia lo que me piden es probar que  $Post_{p2} \rightarrow Post_{p1} \equiv (result = [x^n] \rightarrow x^n - 1 < result \leq x^n)$ .

Esto depende del algoritmo usado para calcular la parte entera de  $x^n$ . Si se usa la función [techo](#), entonces la implicación vale pues  $Post_{p2}$  es literalmente la definición de esa función. Pero si se usa otro algoritmo, por ejemplo la función [piso](#), entonces la implicación no siempre vale.

### 3.7.C. Pregunta C

No necesariamente, depende del algoritmo usado para calcular la parte entera de  $x^n$ .

## 3.8. Ejercicio 8

Notar que  $Pre_{n-esimo1}$  compara con  $<$ , lo cual significa que no pueden haber 2 elementos iguales en la secuencia  $l$ . Por lo tanto, vale que  $Pre_{n-esimo1} \rightarrow Pre_{n-esimo2}$ .

Por otro lado,  $Post_{n-esimo1}$  nos dice que  $result \in l$  y además que está en la posición  $n$ . Debido a que  $Pre_{n-esimo1}$  garantiza que la secuencia  $l$  está ordenada, la forma de obtener el índice de  $result$  definida en  $Post_{n-esimo2}$  en efecto nos va a dar el valor correcto para  $n$ .

Al revés no funciona porque  $Pre_{n-esimo2}$  solo garantiza que los elementos de la secuencia  $l$  sean distintos entre sí, pero eso no implica que la secuencia esté ordenada. Por ejemplo  $\langle 1, 3, 2 \rangle$  satisface  $Pre_{n-esimo2}$  pero no  $Pre_{n-esimo1}$ .

### 3.9. Ejercicio 9

#### 3.9.A. Pregunta A

Dado un número entero, decidir si es par.

```
proc esPar (in n:  $\mathbb{Z}$ , out r: Bool) {  
    Pre {True}  
    Post { $r = True \leftrightarrow n \bmod 2 = 0$ }  
}
```

#### 3.9.B. Pregunta B

Dado un entero  $n$  y uno  $m$ , decidir si  $n$  es un múltiplo de  $m$ .

```
proc esMúltiplo (in n:  $\mathbb{Z}$ , in m:  $\mathbb{Z}$ , out r: Bool) {  
    Pre {True}  
    Post { $r = True \leftrightarrow n \bmod m = 0$ }  
}
```

#### 3.9.C. Pregunta C

Dado un número real, devolver su inverso multiplicativo.

```
proc inversoMultiplicativo (in x:  $\mathbb{R}$ , out r:  $\mathbb{R}$ ) {  
    Pre { $x \neq 0$ }  
    Post { $r = 1/x$ }  
}
```

#### 3.9.D. Pregunta D

Dada una secuencia de caracteres, obtener de ella solo los que son numéricos (con todas sus apariciones sin importar el orden de aparición).

```
proc subseqDeNumeros (in s:  $seq\langle Char \rangle$ , out r:  $seq\langle Char \rangle$ ) {  
    Pre {True}  
    Post {( $\forall c : Char$ )( $\#apariciones(r, c) = \text{if } '0' \leq c \leq '9' \text{ then } \#apariciones(s, c) \text{ else } 0$  fi)}}  
}
```

#### 3.9.E. Pregunta E

Dada una secuencia de reales, devolver la secuencia que resulta de duplicar sus valores en las posiciones impares.

```
proc duplicarPosicionesImpares (in s:  $seq\langle \mathbb{R} \rangle$ , out r:  $seq\langle \mathbb{R} \rangle$ ) {  
    Pre {True}  
    Post {|r| = |s|  $\wedge (\forall i : \mathbb{Z})(0 \leq i < |r| \longrightarrow_L r[i] = \text{if } i \bmod 2 = 0 \text{ then } s[i] \text{ else } s[i] * 2 \text{ fi})$ }  
}
```

### 3.9.F. Pregunta F

Dado un número entero, listar todos sus divisores positivos (sin duplicados).

```
proc divisoresPositivos (in n:  $\mathbb{Z}$ , out r:  $seq(\mathbb{Z})$ ) {  
    Pre {True}  
    Post { $(\forall k : \mathbb{Z})((k > 0 \wedge n \bmod k = 0 \leftrightarrow k \in r) \wedge (\#apariciones(r, k) \leq 1))$ }  
}
```

## 3.10. Ejercicio 10

### 3.10.A. Pregunta A

Tiene sentido la pregunta y la respuesta es que no, 4 no es múltiplo de 0 pues  $\nexists n \in \mathbb{Z} \mid 4 = 0 * n$ .

### 3.10.B. Pregunta B

Sí, debería ser una entrada válida pero no lo es en la especificación dada, pues la precondition pide  $m \neq 0$ .

### 3.10.C. Pregunta C

```
proc esMultiplo] (in n, m:  $\mathbb{Z}$ , out result: Bool) {  
    Pre {True}  
    Post {result = if m = 0 then n = 0 else n mod m = 0 fi}  
}
```

### 3.10.D. Pregunta D

La precondition original es más fuerte pues  $m \neq 0 \rightarrow True$  es una tautología.