



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	jbekenstein@dc.uba.ar



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>

1. Práctica 3

1.1. Ejercicio 1

1.1.1. Pregunta A

El problema es que la post condición 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}  
}
```

1.1.2. Pregunta B

El problema es que se indefine 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]))}  
}
```

1.1.3. Pregunta C

El problema es que en la post condición 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)}  
}
```

1.2. Ejercicio 2

1.2.1. Pregunta A

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

1.2.2. 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 pre condición: $\min_suma(l) \leq suma \leq \max_suma(l) \leftrightarrow 0 \leq 2 \leq 3$ pero no existen elementos en l que sumados den exactamente 2.

1.2.3. Pregunta C

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

1.3. Ejercicio 3

1.3.1. Pregunta A

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

1.3.2. 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) $l = \langle 0, 0, 0, 0, 0, 0 \rangle \rightarrow result \in \{0, 1, 2, 3, 4, 5\}$

1.3.3. 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) $l = \langle 0, 0, 0, 0, 0, 0 \rangle \rightarrow result = 0$

1.3.4. Pregunta D

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

1.4. Ejercicio 4

1.4.1. 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 $a \geq 0$).

1.4.2. Pregunta B

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

1.4.3. Pregunta C

Correcta.

1.4.4. Pregunta D

Correcta.

1.4.5. 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 post condición sea *True* sin importar si efectivamente $result = b - 1$ como debería ser según la especificación. Pasa lo mismo de forma análoga cuando $a < 0$.

1.4.6. Pregunta F

Correcta.

1.5. Ejercicio 5

1.5.1. Pregunta A

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

1.5.2. Pregunta B

- $x = 0.5 \rightarrow result = 0.5^2 = 0.25 \not\geq 0.5$
- $x = 1 \rightarrow result = 1^2 = 1 \not\geq 1$
- $x = -0.2 \rightarrow result = (-0.2)^2 = 0.04 > -0.2$
- $x = -7 \rightarrow result = (-7)^2 = 49 > -7$

1.5.3. Pregunta C

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