



DEPARTAMENTO
DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Práctica 3

1er cuatrimestre 2022

Algoritmos y Estructuras de Datos 1

Integrante	LU	Correo electrónico
Yago Pajariño	546/21	ypajarino@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>

Índice

3. Práctica 3	2
3.1. Ejercicio 1	2
3.2. Ejercicio 2	2
3.3. Ejercicio 3	2
3.4. Ejercicio 4	3
3.5. Ejercicio 5	3
3.6. Ejercicio 6	3
3.7. Ejercicio 7	3
3.8. Ejercicio 8	3
3.9. Ejercicio 9	4
3.10. Ejercicio 10	4
3.11. Ejercicio 11	4
3.12. Ejercicio 12	5
3.13. Ejercicio 13	5
3.14. Ejercicio 14	5
3.15. Ejercicio 15	6
3.16. Ejercicio 16	7
3.17. Ejercicio 17	7
3.18. Ejercicio 18	8
3.19. Ejercicio 19	8
3.20. Ejercicio 20	8
3.21. Ejercicio 21	8
3.22. Ejercicio 22	8

3. Práctica 3

3.1. Ejercicio 1

- (a) La postcondición se indefine si $0 \leq result < |l|$

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

- (b) Se indefine con $i = 0$

```
proc progresiónGeométricaFactor2 (in l: seq⟨ℤ⟩, out result: Bool) {  
  Pre {true}  
  Post {result = true ⇔ ((∀i : ℤ)(0 ≤ i < |l| - 1 →L l[i + 1] = 2.l[i]))}  
}
```

- (c) No se define "x"

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

3.2. Ejercicio 2

- (a) $l = seq⟨1, 2, 3⟩$ y $suma = 7$

- (b) El problema con los límites es que no determina si un valor intermedio resulta producto de la suma de un subconjunto de elementos de l . Ej. $l = seq⟨1, 3⟩$ y $suma = 2$

- (c)

```
proc elementosQueSumen (in l: seq⟨ℤ⟩, in suma: ℤ, out result: seq⟨ℤ⟩) {  
  Pre {(∃m : seq⟨ℤ⟩)(esSubseq(m, l) ∧L ∑i=0|m|-1 l[i] = suma)}  
  Post {(∀x : ℤ)(x ∈ result → #apariciones(x, result) ≤ #apariciones(x, l)) ∧ suma = ∑i=0|result|-1 result[i]}  
}  
  
pred esSubseq (s: seq⟨T⟩, t: seq⟨T⟩) {  
  (∀elem : T)(#apariciones(s, elem) ≤ #apariciones(t, elem))  
}
```

3.3. Ejercicio 3

- (a)
a) 0
b) $\{-1, 1\}$
c) $\{-\sqrt{27}, \sqrt{27}\}$
- (b)
a) 3
b) $\{0, 3\}$
c) $\{0, 1, 2, 3, 4, 5\}$
- a) 3
b) 0
c) 0
- (c) Tienen la misma salida en secuencias sin elementos repetidos.

3.4. Ejercicio 4

- (a) Incorrecta. No se pueden cumplir ambas partes de la conjunción
- (b) Incorrecta. No contempla el caso $a = 0$
- (c) Correcta
- (d) Correcta
- (e) Incorrecta. La implicación junto con la disjunción permite que cualquier valor de result haga verdadera la postcondición.
- (f) Correcta

3.5. Ejercicio 5

- (a) El algoritmo devuelve el valor 9. Hace verdadera la postcondición.
- (b) En $x \in \{0, 1\}$ no cumple la postcondición, en el resto sí.
- (c) $\text{Pre}\{x > 1\}$

3.6. Ejercicio 6

- (a) $P3 \rightarrow P1 \rightarrow P2$
- (b) $Q3 \rightarrow Q1 \rightarrow Q2$
- (c)
 - a) $r = x^2 + 1$
 - b) $r = x^2 + 2$
- (d)
 - a) Si
 - b) No
 - c) Si
 - d) No
 - e) Si
 - f) No
 - g) No
 - h) No
- (e) Se debe cumplir que las precondiciones sean más fuertes y las postcondiciones más débiles.

3.7. Ejercicio 7

1. $(x \neq 0) \rightarrow (\neg(n \leq 0) \vee (x \neq 0))$ Por la regla de la implicación.
2. Sí, la postcondición de P1 es verdadera.
3. No pues $P1 \rightarrow P2$ pero no viceversa. P1 podría recibir valores $n > 0$, no implementados por a .

3.8. Ejercicio 8

Es cierto que todo algoritmo que cumpla con n-esimo1 también cumple con n-esimo2 pues $\text{pre1} \rightarrow \text{pre2}$ y $\text{post1} \rightarrow \text{post2}$, pero no al revés.

3.9. Ejercicio 9

- (a) `proc esPar (in x: \mathbb{Z} , out result: Bool) {`
 `Pre {True}`
 `Post {result = true \iff (x mód 2 = 0)}`
`}`
- (b) `proc esMultiplo (in n: \mathbb{Z} , in m: \mathbb{Z} , out result: Bool) {`
 `Pre {True}`
 `Post {result = true \iff (($\exists k : \mathbb{Z}$)(n = m.k))}`
`}`
- (c) `proc inverso (in x: \mathbb{R} , out result: \mathbb{R}) {`
 `Pre {x \neq 0}`
 `Post {result = $\frac{1}{x}$ }`
`}`
- (d) `proc numericos (in l: seq(Char), out result: seq(Char)) {`
 `Pre {True}`
 `Post {($\forall c : \text{Char}$)(c \in digitos \longrightarrow_L #apariciones(l, c) = #apariciones(result, c)) \wedge (c \in result \longrightarrow c \in digitos)}`
`}`
 digitos = `<'0','1','2','3','4','5','6','7','8','9'>`
- (e) `proc duplicaPosicionesImpares (in l: seq(\mathbb{R}), out result: seq(\mathbb{R})) {`
 `Pre {True}`
 `Post {($\forall i : \mathbb{Z}$)(0 \leq i < |l| \longrightarrow_L ((i mód 2 = 1 \wedge result[i] = 2.l[i]) \vee (i mód 2 = 0 \wedge result[i] = l[i]))}`
`}`
- (f) `proc getDivisores (in x: \mathbb{Z} , out result: seq(\mathbb{Z})) {`
 `Pre {True}`
 `Post {($\forall y : \mathbb{Z}$)(if x mód y = 0 \wedge y > 0 then #apariciones(result, y) = 1 else y \notin result fi)}`
`}`

3.10. Ejercicio 10

- (a) Sí tiene sentido pues tanto 4 como 0 son números enteros. La respuesta es que 4 NO es múltiplo de 0 pues $\neg \exists k \in \mathbb{Z} : 4 = 0.k$
- (b) Debería ser una entrada válida. En la especificación no lo es.
- (c) Ver 9.b
- (d) La nueva precondition {true} es más debil que {m \neq 0}

3.11. Ejercicio 11

- (a) No
- (b) Sí
- (c) Ver 9.e
- (d) La nueva postcondición es más fuerte que la anterior.

3.12. Ejercicio 12

```

proc getBinario (in x:  $\mathbb{Z}$ , out result:  $seq\langle\mathbb{Z}\rangle$ ) {
  Pre  $\{x > 0\}$ 
  Post  $\{x = \sum_{i=0}^{|result|-1} result[i].2^{|result|-i-1}\}$ 
}

```

3.13. Ejercicio 13

Sí, en ambos la precondition es demasiado restrictiva. Se está sobreespecificando.

3.14. Ejercicio 14

- (a)

```

proc sumaDeFactoresPrimos (in x:  $\mathbb{Z}$ , out res:  $\mathbb{Z}$ ) {
  Pre  $\{x > 0\}$ 
  Post  $\{res = \sum_{i=2}^{x-1} \text{if } (esPrimo(i) \wedge x \text{ mód } i = 0) \text{ then } i \text{ else } 0 \text{ fi}\}$ 
}

```
- (b)

```

proc esPerfecto (in x:  $\mathbb{Z}$ , out res: Bool) {
  Pre  $\{x > 0\}$ 
  Post  $\{res = \text{true} \iff x = (\sum_{i=1}^{x-1} \text{if } x \text{ mód } i = 0 \text{ then } i \text{ else } 0 \text{ fi})\}$ 
}

```
- (c)

```

pred sonCoprimos (n,m:  $\mathbb{Z}$ ) {
   $1 = \sum_{i=1}^{n+m} \text{if } (n \text{ mód } i = 0 \wedge m \text{ mód } i = 0) \text{ then } 1 \text{ else } 0 \text{ fi}$ 
}

proc menorCoprimo (in n:  $\mathbb{Z}$ , out m:  $\mathbb{Z}$ ) {
  Pre  $\{n > 0\}$ 
  Post  $\{m > 1 \wedge sonCoprimos(n, m) \wedge (\forall i : \mathbb{Z})(1 \leq i < m \longrightarrow_L \neg sonCoprimos(n, i))\}$ 
}

```
- (d)

```

proc descomposicionEnPrimos (in x:  $\mathbb{Z}$ , out res:  $seq\langle\mathbb{Z} \times \mathbb{Z}\rangle$ ) {
  Pre  $\{x > 0\}$ 
  Post  $\{(x = \sum_{i=0}^{|res|-1} res[i]_0^{res[i]_1})$ 
 $\wedge$ 
 $(\forall i : \mathbb{Z})(0 \leq i < |res| \longrightarrow_L (esPrimo(res[i]_0) \wedge res[i]_1 \geq 1))$ 
 $\wedge$ 
 $(\forall j : \mathbb{Z})(0 \leq j < |res| - 1 \longrightarrow_L res[j]_0 < res[j+1]_0)\}$ 
}

```
- (e)

```

proc diferenciaEntreExtremos (in s:  $seq\langle\mathbb{R}\rangle$ , out res:  $\mathbb{R}$ ) {
  Pre  $\{|s| \geq 2\}$ 
  Post  $\{(\exists i, j : \mathbb{R})(i, j \in s \wedge (\forall a : \mathbb{R})(a \in s \longrightarrow_L i \leq a \leq j) \wedge res = j - i)\}$ 
}

```

(f) **aux cantQueDivide** ($x: \mathbb{Z}, l: seq\langle \mathbb{Z} \rangle$) : $\mathbb{Z} = \sum_{i=0}^{|l|-1} \text{if } l[i] \bmod x = 0 \text{ then } 1 \text{ else } 0 \text{ fi}$;

```

proc divideAMasElementos (in l: seq⟨ℤ⟩, out res: ℤ) {
  Pre { |l| > 0 }
  Post { res ∈ l ∧ (∀i : ℤ)(0 ≤ i < |l| →L cantQueDivide(l[i], l) ≤ cantQueDivide(res, l)) }
}

```

3.15. Ejercicio 15

- (a) **proc nEsimaAparicion** (in l: seq⟨ℝ⟩, in e: ℝ, out result: ℤ) {
- ```

 Pre { e ∈ l ∧ n ≥ 1 }
 Post { (l[result] = e) ∧ (n - 1 = ∑_{i=0}^{result-1} if l[i] = e then 1 else 0 fi) }
}

```
- (b) **proc esSubcadena** (in s: seq⟨ℝ⟩, in t: seq⟨ℝ⟩, out result: Bool) {
- ```

  Pre { true }
  Post { |s| ≤ |t| ∧ ((∃i : ℤ)(0 ≤ i < |t| ∧ (∃j : ℤ)(0 ≤ j < i ∧ s == subseq(t, j, i)))) }
}

```
- (c) **proc estaIncluida** (in s: seq⟨ℤ⟩, in t: seq⟨ℤ⟩, out res: Bool) {
- ```

 Pre { true }
 Post { res = True ⇔ (∀i : ℤ)(0 ≤ i < |s| →L #apariciones(s[i], t) ≥ #apariciones(s[i], s)) }
}

```
- (d) **proc mezclarOrdenado** (in s: seq⟨ℤ⟩, in t: seq⟨ℤ⟩, out res: seq⟨ℤ⟩) {
- ```

  Pre { estaOrdenada(s) ∧ estaOrdenada(t) }
  Post { |res| = |s| + |t|
    ∧
    estaOrdenada(res)
    ∧
    (∀x : ℤ)(#apariciones(x, res) = #apariciones(x, s) + #apariciones(x, t)) }
}

```
- (e) **proc interseccionSinRepetidos** (in s: seq⟨ℤ⟩, in t: seq⟨ℤ⟩, out res: seq⟨ℤ⟩) {
- ```

 Pre { True }
 Post { estanTodos(s, res) ∧ estanTodos(t, res) ∧ noHayRepetidos(res) }
}

```
- pred estanTodos** (l: seq⟨ℤ⟩, r: seq⟨ℤ⟩) {
- ```

  (∀i : ℤ)(0 ≤ i < |l| →L (∃j : ℤ)r[j] = l[i])
}

```
- pred noHayRepetidos** (r: seq⟨ℤ⟩) {
- ```

 (∀x : ℤ)(#apariciones(x, r) ≤ 1)
}

```
- (f) **proc interseccion** (in s, t: seq⟨T⟩, out res: seq⟨T⟩) {
- ```

  Pre { True }
  Post { (∀elem : T)(#apariciones(res, elem) = min(#apariciones(s, elem), #apariciones(t, elem))) }
}

```

3.16. Ejercicio 16

1. **proc** `cantApariciones` (in `l: String`, out `res: seq⟨Char × ℤ⟩`) {
 Pre {*True*}
 Post {*mismosElementosEnDict*(*l*, *res*) ∧ *sinKeysRepetidos*(*res*) ∧
 ($\forall i : \mathbb{Z})(0 \leq i < |res| \rightarrow_L res[i]_1 = \#apariciones(l, res[i]_0)$)}
 }

- pred** `mismosElementosEnDict` (*s: seq⟨T⟩*, *d: seq⟨T × U⟩*) {
 ($\forall elem : T)((elem \in s \wedge existeKey(elem, d)) \vee (elem \notin s \wedge \neg existeKey(elem, d)))$)
 }

- pred** `existeKey` (*elem: T*, *d: seq⟨T × U⟩*) {
 ($\exists i : \mathbb{Z})(0 \leq i < |d| \wedge_L d[i]_0 = elem)$)
 }

- pred** `sinKeysRepetidos` (*d: seq⟨T × U⟩*) {
 ($\forall i, j : \mathbb{Z})((0 \leq i, j < |d| \wedge i \neq j) \rightarrow_L d[i]_0 \neq d[j]_0)$)
 }

- proc** `prefijos` (in *s: seq⟨T⟩*, out *res: seq⟨seq⟨T⟩⟩*) {
 Pre {*True*}
 Post { $|res| = |s| + 1 \wedge_L (\forall i : \mathbb{Z})(0 \leq i < |res| \rightarrow_L res[i] = subseq(s, 0, i))$ }
 }

3. **proc** `secuenciaConElMayorValor` (in *s: seq⟨seq⟨ℤ⟩⟩*, out *res: seq⟨ℤ⟩*) {
 Pre { $|s| > 0$ }
 Post { $res \in s \wedge (\forall l : seq⟨\mathbb{Z}⟩)(l \in s \rightarrow contieneElMayorValorEntre(res, l))$ }
 }

- pred** `contieneElMayorValorEntre` (*s, t: seq⟨ℤ⟩*) {
 ($\forall k : \mathbb{Z})(k \in t \rightarrow (\exists l : \mathbb{Z})(l \in s \wedge l \geq k))$)
 }

4. **proc** `interseccionMultiple` (in *ls: seq⟨seq⟨ℝ⟩⟩*, out *l: seq⟨ℝ⟩*) {
 Pre {*True*}
 Post {*noHayRepetidos*(*l*) ∧ ($\forall x : \mathbb{R})(\exists i : \mathbb{Z})(0 \leq i < |ls| \wedge_L \#apariciones(x, ls[i]) > 0) \rightarrow_L x \in l$)}
 }

5. TODO

3.17. Ejercicio 17

1. No es correcta. No guarda el valor inicial de a,b en la precondition.
2. No es correcta. c es una variable in.
3. Es correcta.
4. Es correcta.

3.18. Ejercicio 18

1. No es correcta. No guarda el valor inicial de l.
2. No es correcta. No saca el primer elemento a l.
3. No es correcta. Si bien saca un elemento a l, cualquier elemento borrado de l cumple con la especificación.
4. Es correcta.
5. Es correcta.

3.19. Ejercicio 19

TODO

3.20. Ejercicio 20

Dada una secuencia y un entero i, reemplaza los elementos de la secuencia por el i-ésimo y reemplaza al e-ésimo elemento de la secuencia por el primero.

3.21. Ejercicio 21

1. No especifica asignación en las posiciones impares.
2. No especifica la longitud de l.
3. l debería ser una variable in en lugar de inout.

3.22. Ejercicio 22

TODO