



## Ejercicio 1.

Sea la siguiente especificación del problema de copiar una secuencia de enteros:

```
proc copiarSecuencia (in s: seq(Z), out result: seq(Z)) {
    Pre {True}
    Post {s = result}
}
```

Sea la siguiente implementación incompleta de la función copiarSecuencia:

```
vector<int> copiarSecuencia(vector<int> s) {
    vector<int> r;
    int i = 0;
    while(i < s.size()) {
        ...
    }
    return r;
}
```

Completar el programa (i.e. escribir el cuerpo del **while**) de forma que cumpla el siguiente invariante de ciclo:

$$I = (0 \leq i \leq |s| \wedge |r| = i) \wedge_L (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] = r[j])$$

## Ejercicio 2.

Sea la siguiente especificación:

```
proc incSecuencia (inout s: seq(Z)) {
    Pre {s = S0}
    Post {|s| = |S0| ∧L (∀i : Z)(0 ≤ i < |s| →L s[i] = S0[i] + 1)}
}
```

Sea la siguiente implementación incompleta de la función incSecuencia:

```
void incSecuencia(vector<int> &a) {
    int i = 0;
    while(...) {
        ...
    }
}
```

Completar el programa (i.e. escribir el cuerpo del **while** y su guarda) de forma que cumpla el siguiente invariante de ciclo:

$$I = 0 \leq i \leq |s| \wedge (\forall j : \mathbb{Z})(0 \leq j < i \rightarrow_L s[j] = S_0[j] + 1)$$

## Ejercicio 3.

Sea la siguiente especificación del problema de retornar la cantidad de apariciones de un elemento en una secuencia de enteros:

```
proc cantApariciones (in s: seq(Z), in e: Z, out result: Z) {
    Pre {True}
    Post {result = #apariciones(s, e)}
}
```

Sea la siguiente implementación incompleta de la función cantApariciones:

```
int cantApariciones(vector<int> s, int e) {
    int r = 0;
    for(int i=0; ... ; ...) {
        ...
    }
    return r;
}
```

Completar el programa (i.e. escribir el cuerpo y la declaración del **for**) de forma que cumpla el siguiente invariante de ciclo:

$$I = 0 \leq i \leq |s| \wedge_L r = \#apariciones(subseq(s, 0, i), e))$$

#### Ejercicio 4.

Sea la siguiente especificación de un ciclo:

- $P_c : i = -1 \wedge s = S_0$
- $Q_c : |s| = |S_0| \wedge_L (\forall z : \mathbb{Z})(0 \leq z < |s| \rightarrow_L s[z] = S_0[z]^2)$

Dar dos implementaciones distintas que satisfagan la especificación del ciclo con el siguiente invariante:

$$I = (|s| = |S_0| \wedge -1 \leq i \leq |s| - 1) \wedge_L (\forall j : \mathbb{Z})(0 \leq j \leq i \rightarrow_L s[j] = S_0[j]^2) \wedge (\forall j : \mathbb{Z})(i < j < |s| \rightarrow_L s[j] = S_0[j])$$

#### Ejercicio 5.

Escribir un programa para el siguiente problema que respete la especificación y el invariante dado

```

proc duplicarElementos (inout s : seq(Z)) {
  Pre {s = s0 ∧ |s| mod 2 = 0}
  Post {|s| = |s0| ∧L
    (∀i : Z)(enRango(i, s) →L s[i] = 2 * s0[i])}

```

$$I \equiv (|s| = |s_0| \wedge (0 \leq i \leq |s|/2)) \wedge_L$$

$$subseq(s, 0, |s| - 2 * i) = subseq(s_0, 0, |s_0| - 2 * i) \wedge$$

$$(\forall k : \mathbb{Z})(|s| - 2 * i \leq k < |s| \rightarrow_L s[k] = 2 * s_0[k])$$

#### Ejercicio 6.

Escribir un programa para el siguiente problema que respete la especificación y contenga un ciclo el invariante dado

```

proc dividirPorPromedio (inout s : seq(R)) {
  Pre {s = s0 ∧ |s| mod 2 = 0 ∧ |s| > 0}
  Post {|s| = |s0| ∧L
    (∀i : Z)(enRango(i, s) →L s[i] =  $\frac{s_0[i]}{promedio(s_0)}$ )}
  aux promedio (s : seq(R)) : R =  $\frac{\sum_{i=0}^{|s|-1} s[i]}{|s|}$ ;

```

$$I \equiv (|s| = |s_0| \wedge 0 \leq i \leq \frac{|s|}{2}) \wedge_L$$

$$subseq(s, i, |s| - i) = subseq(s_0, i, |s_0| - i) \wedge$$

$$(\forall k : \mathbb{Z})(0 \leq k < i \rightarrow_L s[k] = \frac{s_0[k]}{promedio(s_0)}) \wedge$$

$$(\forall k : \mathbb{Z})(|s| - i - 1 < k < |s| \rightarrow_L s[k] = \frac{s_0[k]}{promedio(s_0)})$$

#### Ejercicio 7.

Dar un programa que satisfaga la especificación y tenga un ciclo con el invariante:

```

proc armarPiramide (in v : Z, inout l : seq(Z)) {
  Pre {l = L0}
  Post {|L0| = |l| ∧ esPiramide(l, v)}
  pred esPiramide (l : seq(Z), v : Z) {
    (∀j : Z)(0 ≤ j < |l|/2 ⇒L l[j] = v + j) ∧
    (∀j : Z)(|l|/2 ≤ j < |l| ⇒L l[j] = v + |l| - j - 1)
  }
}

```

a. **Invariante:**  $|l| = |L_0| \wedge |l|/2 \leq i \leq |l| \wedge_L ((i = |l|/2 \wedge l = L_0) \vee_L$   
 $(\exists p : seq(\mathbb{Z}))(esPiramide(p, v) \wedge |p| = |l| \wedge_L subseq(p, |l| - i, i) = subseq(l, |l| - i, i)))$

b. **Invariante:**  $|l| = |L_0| \wedge 0 \leq i \leq |l| \wedge_L piramideHastaI(l, i, v)$   
 pred piramideHastaI (l : seq(Z), i : Z, v : Z) {  
 $(\exists p : seq(\mathbb{Z}))(esPiramide(p, v) \wedge |p| = |l| \wedge_L subseq(p, 0, i) = subseq(l, 0, i))$   
 }

#### Ejercicio 8.

Escribir un programa para el siguiente problema que respete la especificación y el invariante dado

```

proc multiplicar (inout s : seq(R)) {
  Pre {s = s0 ∧ |s| mod 4 = 0}
  Post {|s| = |s0| ∧L
    (∀i : Z)(0 ≤ i < |s| →L s[i] = 10 * s0[i])}
}

```

$$I \equiv (|s| = |s_0| \wedge \frac{|s|}{2} \leq i \leq |s|) \wedge_L esPar(i) \wedge$$

$$subseq(s, i, |s|) = subseq(s_0, i, |s_0|) \wedge subseq(s, 0, |s| - i) = subseq(s_0, 0, |s_0| - i) \wedge$$

$$(\forall k : \mathbb{Z})(\frac{|s|}{2} \leq k < i \rightarrow_L s[k] = 10 * s_0[k]) \wedge (\forall k : \mathbb{Z})(|s| - i \leq k < \frac{|s|}{2} \rightarrow_L s[k] = 10 * s_0[k])$$

**Ejercicio 9.**

Escribir un programa para el siguiente problema que respete la especificación y el invariante dado

```

proc cerearYsumar (inout s : seq⟨ℤ⟩, inout suma : ℤ) {
  Pre {s = s0 ∧ |s| mod 8 = 0}
  Post {|s| = |s0| ∧L ((∀i : ℤ)(enRango(i, s) →L s[i] = 0) ∧ suma = ∑i=0|s|-1 s0[i])}
}

I ≡ (|s| = |s0| ∧ 0 ≤ i ≤ |s|/4) ∧L
(subseq(s, 2 * i, |s| - 2 * i) = subseq(s0, 2 * i, |s0| - 2 * i) ∧ (∀k : ℤ)(0 ≤ k < 2 * i →L s[k] = 0) ∧
(∀k : ℤ)(|s| - 2 * i ≤ k < |s| →L s[k] = 0) ∧ suma = ∑j=02*i-1 s0[j] + ∑j=|s|-2*i|s|-1 s0[j])

```

## RESOLUCIONES.

**Ejercicio 1.** programa:

```
1 vector<int> copiarSecuencia(vector<int> s){
2     vector<int> r;
3     int i=0;
4     while(i<s.size()){
5         r.push_back(s[i]);
6         i++;
7     }
8     return r;
9 }
```

```
1 vector<int> copiarSecuencia(vector<int> s){
2     vector<int> r;
3     int i=0;
4     while(i<s.size()){
5         r.push_back(s[i]);
6         i++;
7     }
8     return r;
9 }
```