



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 ≡ (|s| = |s0| ∧ (0 ≤ i ≤ |s|/2)) ∧L
    subseq(s, 0, |s| - 2 * i) = subseq(s0, 0, |s0| - 2 * i) ∧
    (∀k : Z)(|s| - 2 * i ≤ k < |s| →L s[k] = 2 * s0[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] =  $\frac{s_0[i]}{\text{promedio}(s_0)}$ )}
  aux promedio (s : seq(R)) : R =  $\frac{\sum_{i=0}^{|s|-1} s[i]}{|s|}$ ;
  I ≡ (|s| = |s0| ∧ 0 ≤ i ≤  $\frac{|s|}{2}$ ) ∧L
    subseq(s, i, |s| - i) = subseq(s0, i, |s0| - i) ∧
    (∀k : Z)(0 ≤ k < i →L s[k] =  $\frac{s_0[k]}{\text{promedio}(s_0)}$ ) ∧
    (∀k : Z)(|s| - i - 1 < k < |s| →L s[k] =  $\frac{s_0[k]}{\text{promedio}(s_0)}$ )
```

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| = |L0| ∧ |l|/2 ≤ i ≤ |l| ∧L ((i = |l|/2 ∧ l = L0) ∨L
  (∃p : seq(Z))(esPiramide(p, v) ∧ |p| = |l| ∧L subseq(p, |l| - i, i) = subseq(l, |l| - i, i)))

b. Invariante: |l| = |L0| ∧ 0 ≤ i ≤ |l| ∧L piramideHastaI(l, i, v)
  pred piramideHastaI (l : seq(Z), i : Z, v : Z) {
    (∃p : seq(Z))esPiramide(p, v) ∧ |p| = |l| ∧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 ≡ (|s| = |s0| ∧  $\frac{|s|}{2} \leq i \leq |s|$ ) ∧L esPar(i) ∧
  subseq(s, i, |s|) = subseq(s0, i, |s0|) ∧ subseq(s, 0, |s| - i) = subseq(s0, 0, |s0| - i) ∧
  (∀k : Z)( $\frac{|s|}{2} \leq k < i$  →L s[k] = 10 * s0[k]) ∧ (∀k : Z)(|s| - i ≤ k <  $\frac{|s|}{2}$  →L s[k] = 10 * s0[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 .

```
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 }
```

.

Ejercicio 2 .

```
1 vector<int> incSecuencia(vector<int> &s){
2     int i=0;
3     while(i<s.size()){
4         s[i]++;
5         i++;
6     }
7 }
```

.

Ejercicio 3 .

```
1 int cantApariciones(vector<int> s, int e){
2     int r=0;
3     for(int i=0; i<s.size();i++){
4         r=r+(s[i]==e);
5     }
6     return r;
7 }
```

.

Ejercicio 4 .

```
1 void raizConEsei(vector<int> &s){
2     int i=-1;
3     vector<int> s0=s;
4     while( i<s.size()-1){
5         i++;
6         s[i]=sqrt(s0[i]);
7     }
8     return;
9 }
```

.

Ejercicio 5 .

```
1 void duplicarElementos(vector<int> &s){
2     int i=0;
3     vector<int> s0=s;
4     while( i<s.size()/2){
```

```

5         s[s.size()-2*i-1]=2*s0[s0.size()-2*i-1];
6         s[s.size()-2*i-2]=2*s0[s0.size()-2*i-2];
7         i++;
8     }
9     return;
10 }

```

Ejercicio 6 .

```

1 void dividirPorPromedio(vector<float> &s){
2     int i=0;
3     vector<float> s0=s;
4     int p=promedio(s0);
5     while( i<s.size()/2){
6         s[i]=s0[i]/p;
7         s[s.size()-i-1]=s0[s0.size()-i-1]/p;
8         i++;
9     }
10    return;
11 }
12
13 float promedio(vector<float> &s){
14     float res=0;
15     for(int i=0;i<s.size();i++){
16         res+=s[i];
17     }
18     res/=s.size();
19     return res;
20 }

```

Ejercicio 7

a) primer I.

```

1 void armarPiramide(int v,vector<int> &s){
2     int i=0;
3     vector<float> s0=s;
4     while( i<s.size()/2){
5         s[i]=i+v;
6         s[s.size()-i-1]=v+s.size()-i-1;
7         i++;
8     }
9     return;
10 }

```

b) segundo I.

```

1 void armarPiramide(int v,vector<int> &s){
2     int i=0;
3     while(i<s.size()/2){
4         s[i]=i+v;
5         i++;
6     }

```

```

7     while(i<s.size()){
8         s[i]=v+s.size()-i-1;
9         i++;
10    }
11    return;
12 }

```

Ejercicio 8 .

```

1 void multiplicar(vector<int> &s){
2     int i=s.size()/2;
3     vector<float> s0=s;
4     while( i<s.size()){
5         s[s.size()-i-1]=10*s0[s0.size()-i-1];
6         s[i]=10*s0[i];
7         i++;
8     }
9     return;
10 }

```