**Ejercicio 1.** Dado el siguiente ciclo con sus correspondientes pre y postcondición:

$P_c : \{|s| > 0 \wedge i = 1 \wedge r = \textbf{true}\}$

```
while (i < s.size()) do
  r := r && (s[i - 1] == s[i]);
  i := i + 1
endwhile
```

$Q_c : \{r = \textbf{true} \leftrightarrow (\exists x : \mathbb{Z})(\forall k : \mathbb{Z})(0 \leq k < |s| \rightarrow_L s[k] = x)\}$

proponer un invariante $I$ para el ciclo y demostrar que se verifican los siguientes puntos del teorema del invariante:

a) $(I \wedge \neg B) \Rightarrow Q_c$

b) $\{I \wedge B\}$ $\langle$cuerpo del ciclo$\rangle$ $\{I\}$

**Ejercicio 2.** La relación de *orden lexicográfico* se escribe "$\sqsubset$" y corresponde al orden en el que aparecen las palabras en el diccionario. Por ejemplo, sabemos que ALA $\sqsubset$ ALADO $\sqsubset$ ARCO $\sqsubset$ BALA. En este ejercicio trabajaremos con el orden lexicográfico sobre secuencias de enteros. Más precisamente, se pide:

a) Especificar el predicado auxiliar $\mathsf{pred\ menorLex}(s : \mathsf{seq}(\mathbb{Z}), t : \mathsf{seq}(\mathbb{Z}))$, que es verdadero si y sólo si $s$ es lexicográficamente menor que $t$, es decir $s \sqsubset t$. Más precisamente, $s \sqsubset t$ si y sólo si, para algún entero $k$, las secuencias $s$ y $t$ coinciden en un prefijo de longitud $k$ y se da una de las dos opciones siguientes:

- o bien la secuencia $s$ tiene exactamente $k$ elementos (con índices $0, \ldots, k-1$) y $t$ es estrictamente más larga,
- o bien el valor de $s$ en la posición $k$ es estrictamente menor que el valor de $t$ en esa posición.

Por ejemplo, $\langle\rangle \sqsubset \langle 1\rangle \sqsubset \langle 1,2,3\rangle \sqsubset \langle 1,4,3\rangle \sqsubset \langle 1,4,5\rangle \sqsubset \langle 1,5\rangle \sqsubset \langle 2\rangle$.

b) Especificar el problema que recibe como entrada dos secuencias de enteros $s$ y $t$, donde $t$ debe ser no vacía, y modifica la secuencia $s$ de tal modo que la secuencia modificada $s'$ es de igual longitud que $s$ y lexicográficamente menor que $t$. Además, $s'$ debe estar "lo más cerca posible" de $s$, es decir, se debe minimizar la *distancia de Manhattan*[1] entre $s$ y $s'$. Por último, se debe devolver el entero $d$ que representa la distancia entre $s$ y $s'$.

Por ejemplo:

- Si $s = \langle 1,2,3\rangle$ y $t = \langle 1,2,3,4\rangle$, cabe notar que $s \sqsubset t$ y por lo tanto la única respuesta posible es $s' = s$ y $d = 0$.
- Si $s = \langle 1,2,5\rangle$ y $t = \langle 1,2,3,4\rangle$ una respuesta posible es $s' = \langle 0,2,5\rangle$ con $d = 1$. Otra respuesta posible es $s' = \langle 1,1,5\rangle$, también con $d = 1$ (como no podría ser de otra manera, ya que la respuesta debe minimizar la distancia).
- Si tuviéramos $s = \langle 1,2,3\rangle$ y $t = \langle\rangle$, **no habría manera** de modificar $s$ para que sea lexicográficamente menor que $t$.

**Nota (1):** Si $t$ es vacía el problema no tiene solución. Puede asumir sin demostrarlo que, si $t$ es no vacía, siempre se puede encontrar una secuencia $s'$ de igual longitud que $s$ y lexicográficamente menor que $t$.
**Nota (2):** Puede asumir ya definida una función auxiliar $\mathsf{aux\ distancia}(s : \mathsf{seq}(\mathbb{Z}), t : \mathsf{seq}(\mathbb{Z})) : \mathbb{Z}$, que devuelve la distancia de Manhattan entre dos secuencias de enteros, que se asumen de la misma longitud.

[1] La distancia de Manhattan es la suma de las diferencias (en valor absoluto) de los elementos de $s$ y $t$, por ejemplo $\mathsf{distancia}(\langle 4,2,9\rangle, \langle 8,5,7\rangle) = |4-8| + |2-5| + |9-7| = 4+3+2 = 9$.

**Ejercicio 3.** Dados el siguiente programa $S$ en SmallLang y la siguiente especificación:

```
if (s[i] > s[i + 1]) then
  tmp := s[i];
  s[i] := s[i + 1];
  s[i + 1] := tmp
else
  skip
endif
```

pred ordenadaHasta $(s : seq\langle\mathbb{Z}\rangle,\ i : \mathbb{Z})$ {
$\quad (\forall j : \mathbb{Z})(1 \leq j + 1 < i \longrightarrow_L s[j] \leq s[j + 1])$
}

proc ajustar (inout $s$: $seq\langle\mathbb{Z}\rangle$, in $i$: $\mathbb{Z}$) {
$\quad$ Pre $\{(s = S_0 \land 1 \leq i + 1 < |s|) \land_L \texttt{ordenadaHasta}(s, i)\}$
$\quad$ Post $\{(|s| = |S_0| \land 1 \leq i + 1 < |s|) \land_L \texttt{ordenadaHasta}(s, i + 1)\}$
}

a) Calcular la precondición más débil del programa S con respecto a la postcondición de la especificación: *wp(S; Post).*

b) ¿El programa es correcto con respecto a la especificación? En caso de que lo sea, demostrarlo. En caso de que no lo sea, justificar detalladamente por qué el programa no es correcto y qué parte de la demostración fallaría.

**Ejercicio 4.**

Sea el siguiente programa y su especificación:

```
     void f(vector <int>& s1, vector <int>& s2){
L1:      int i = 0;
L2:      int a = 0;
L3:      int b = 0;

L4:      while (i < s1.size()) {
L5:          a = s1[i];

L6:          if (i >= s2.size()) {
L7:              b = 0;
             } else {
L8:              b = s2[i];
             }

L9:          s1[i] = a + b;

L10:         if (i < s2.size()) {
L11:             if (a - b > 0) {
L12:                 s2[i] = b - a;
                 } else {
L13:                 s2[i] = a - b;
                 }
             }
L14:         i++;
         }
     }
```

proc f (inout s1: $seq\langle\mathbb{Z}\rangle$, inout s2: $seq\langle\mathbb{Z}\rangle$) {
$\quad$ Pre $\{s1_0 = s1 \land s2_0 = s2\}$
$\quad$ Post {
$\quad (\forall i \in \mathbb{Z})$
$\quad ((0 \leq i < |s1| \land 0 \leq i < |s2|) \longrightarrow_L$
$\quad\quad (s1[i] = s1_0[i] + s2_0[i]) \land$
$\quad\quad (s2[i] = abs(s1_0[i] - s2_0[i]))) \land$
$\quad ((i \geq |s1| \land 0 \leq i < |s2|) \longrightarrow_L s2[i] = s2_0[i]) \land$
$\quad ((i \geq |s2| \land 0 \leq i < |s1|) \longrightarrow_L s1[i] = s1_0[i])\}$
}

**Cada caso de test propuesto debe contener la entrada y el resultado esperado.**

a) Describir el diagrama de control de flujo (*control-flow graph*) del programa.

b) Escribir un conjunto de casos de test (o *test suite*) que cubra todas las sentencias. Mostrar qué líneas cubre cada test. Este conjunto de tests ¿cubre todas las decisiones? (Justificar).

c) Escribir un *test* que encuentre el defecto presente en el código (una entrada que cumple la precondición pero tal que el resultado de ejecutar el código no cumple la postcondición).

d) ¿Es posible escribir para este programa un *test suite* que cubra todas las decisiones pero que no encuentre el defecto en el código? En caso afirmativo, escribir el test suite; en caso negativo, justificarlo.