



Trabajo práctico 1

Especificación y WP

14 de septiembre de 2024

Algoritmos y Estructura de Datos

SinGrupo4

Integrante	LU	Correo electrónico
Algañaraz, Franco	1092/22	francoarga10@gmail.com
Illescas, Marcos	390/14	marcosillescas90@gmail.com
Bahamonde, Matias	694/21	matubaham@gmail.com
Marión, Ian Pablo	004/01	ianfrodin@gmail.com



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. Especificación

```
pred habitantesPositivos (in s: seq⟨Ciudad⟩) {
  (∀j : ℤ)(0 ≤ j < |s| →L s[j].habitantes ≥ 0)
}
pred noCiudadesRepetidas (in s: seq⟨Ciudad⟩) {
  (∀i : ℤ)(∀j : ℤ)(0 ≤ i < j < |s| →L s[i].nombre ≠ s[j].nombre)
}
pred matrizCuadrada (in s: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ)(0 ≤ i < |s| →L |s| = |s[i]|)
}
pred matrizPositivaSimetricaConCerosDiagonal (in s: seq⟨seq⟨ℤ⟩⟩) {
  (∀i : ℤ)(∀j : ℤ)(0 ≤ i < j < |s| →L s[i]i = 0 ∧ s[i]j = s[j]i ∧ s[i]j ≥ 0)
}
pred esCamino (in s: seq⟨ℤ⟩, in n: ℤ, in m: ℤ, in A: seq⟨seq⟨ℤ⟩⟩) {
  |s| ≥ 2 ∧ s[0] = n ∧ s[|s| - 1] = m ∧
  (∀i : ℤ)(0 ≤ i < |s| →L 0 ≤ s[i] < |A|) ∧
  (∀j : ℤ)(0 ≤ j < |s| - 1 →L A[s[j]]s[j+1] > 0)
}
```

1.1. grandesCiudades

```
proc grandesCiudades (in ciudades: seq⟨Ciudad⟩) : seq⟨Ciudad⟩
  requiere {habitantesPositivos(ciudades) ∧ noCiudadesRepetidas(ciudades)}
  asegura {|res| = CantidadCiudadesMayor50000(ciudades) ∧
  (∀i : ℤ)(0 ≤ i < |res| →L (∃j : ℤ)(0 ≤ j < |ciudades| ∧L (res[i] = ciudades[j] ∧ ciudades[j].habitantes > 50000))))}
aux CantidadCiudadesMayor50000 (in s: seq⟨Ciudad⟩) : ℤ =
  ∑i=0|s|-1 if s[i].habitantes > 50000 then 1 else 0 fi;
```

1.2. sumaDeHabitantes

```
proc sumaDeHabitantes (in menoresDeCiudades: seq⟨Ciudad⟩, in mayoresDeCiudades: seq⟨Ciudad⟩) : seq⟨Ciudad⟩
  requiere {|menoresDeCiudades| = |mayoresDeCiudades| ∧
  mismosNombres(menoresDeCiudades, mayoresDeCiudades) ∧
  habitantesPositivos(menoresDeCiudades) ∧ habitantesPositivos(mayoresDeCiudades) ∧
  noCiudadesRepetidas(menoresDeCiudades) ∧ noCiudadesRepetidas(mayoresDeCiudades)}
  asegura {|res| = |menoresDeCiudades| ∧
  (∀i : ℤ)(0 ≤ i < |res| →L (∃j : ℤ)(∃k : ℤ)(0 ≤ j < k < |menoresDeCiudad| ∧L
  siCoincidenNombresSumoHabitantes(res, menoresDeCiudades, mayoresDeCiudades, i, j, k))))}
pred mismosNombres (in s: seq⟨Ciudad⟩, in t: seq⟨Ciudad⟩) {
  (∀i : ℤ)(0 ≤ i < |s| →L (∃j : ℤ)(0 ≤ j < |t| ∧L s[i].nombre = t[j].nombre))
}
pred siCoincidenNombresSumoHabitantes (in s: seq⟨Ciudad⟩, in t: seq⟨Ciudad⟩, in r: seq⟨Ciudad⟩, in i: ℤ, in j: ℤ, in k: ℤ) {
  (s[i].nombre = t[j].nombre ∧ s[i].nombre = r[k].nombre) →L s[i].habitantes = t[j].habitantes + r[k].habitantes
}
```

1.3. hayCamino

```
proc hayCamino (in distancias : seq⟨seq⟨ℤ⟩⟩, in desde : ℤ, in hasta : ℤ) : Bool
  requiere {matrizCuadrada(distancias) ∧ matrizPositivaSimetricaConCerosDiagonal(distancias) ∧
  0 ≤ desde < |distancias| ∧ 0 ≤ hasta < |distancias| ∧ desde ≠ hasta}
  asegura {res = true ↔ (∃c : seq⟨ℤ⟩)(esCamino(c, desde, hasta, distancias))}
```

1.4. cantidadCaminosNSaltos

```
proc cantidadCaminosNSaltos (inout conexion: seq⟨seq⟨ℤ⟩⟩, in n: ℤ)
  requiere {n ≥ 1 ∧ matrizCuadrada(conexion) ∧
  matrizPositivaSimetricaConCerosDiagonal(conexion) ∧ unosYCeros(conexion) ∧ conexion = C0}
```

```

asegura { (∃s : seq⟨seq⟨seq⟨Z⟩⟩⟩)(|s| = n ∧ s[0] = C0 ∧
(∀i : Z)(0 ≤ i < n →L matrizCuadrada(s[i]) ∧ matrizPositivaSimetricaConCerosDiagonal(s[i]) ∧
(∀j : Z)(1 ≤ j < n →L esMultiplicacionMatricialDe(s[j], s[j - 1], C0)) ∧ conexion = s[n - 1])) }

pred unosYceros (in s : seq⟨seq⟨Z⟩⟩) {
  (∀i : Z)(∀j : Z)(0 ≤ i < j < |s| →L s[i]j = 0 ∨ s[i]j = 1)
}

pred esMultiplicacionMatricialDe (in A : seq⟨seq⟨Z⟩⟩, in B : seq⟨seq⟨Z⟩⟩, in C : seq⟨seq⟨Z⟩⟩) {
  (∀i : Z)(∀j : Z)(0 ≤ i ≤ j < |A| →L A[i]j = ∑k=0|A|-1 B[i]k * C[k]j)
}

```

1.5. caminoMínimo

```

proc caminoMinimo (in origen : Z, in destino : Z, in distancias : seq⟨seq⟨Z⟩⟩) : seq⟨Z⟩
  requiere { matrizCuadrada(distancias) ∧ matrizPositivaSimetricaConCerosDiagonal(distancias) ∧
0 ≤ origen < |distancias| ∧ 0 ≤ destino < |distancias| ∧ origen ≠ destino }
  asegura { (|res| = 0 ↔ ¬(∃s : seq⟨Z⟩)(esCamino(s, origen, hasta, distancias))) ∨
(esCamino(res, origen, hasta, distancias) ∧ (∀c : seq⟨Z⟩)(esCamino(c, origen, destino, distancias) →
distanciaTotal(res, distancias) ≤ distanciaTotal(c, distancias))) }
  aux distanciaTotal (in s : seq⟨Z⟩, in A : seq⟨seq⟨Z⟩⟩) : Z = ∑i=0|c|-2 A[c[i]]c[i+1];

```

2. Demostraciones de correctitud

2.1. Demostrar que la implementación es correcta con respecto a la especificación.

```

proc poblacionTotal (in ciudades : seq⟨Ciudad⟩) : Z
  requiere { (∃j : Z)(0 ≤ j < |ciudades| ∧L ciudades[j].habitantes > 50000) ∧
(∀k : Z)(0 ≤ k < |ciudades| →L ciudades[k].habitantes ≥ 0) ∧
(∀m : Z)(∀n : Z)(0 ≤ m < n < |ciudades| →L ciudades[m].nombre ≠ ciudades[n].nombre) }
  asegura { res = ∑i=0|ciudades|-1 ciudades[i].habitantes }

```

```

S1 : res := 0
S2 : i := 0
while (i < ciudades.length) do
S3 : res := res + ciudades[i].habitantes
S4 : i := i + 1
endwhile

```

```

P ≡ A ∧ B ∧ C
A ≡ (∃l : Z)(0 ≤ l < |ciudades| ∧L ciudades[l].habitantes > 50000)
B ≡ (∀k : Z)(0 ≤ k < |ciudades| →L ciudades[k].habitantes ≥ 0)
C ≡ (∀m : Z)(∀n : Z)(0 ≤ m < n < |ciudades| →L ciudades[m].nombre ≠ ciudades[n].nombre)

```

$Pc \equiv res = 0 \wedge i = 0 \wedge A \wedge B \wedge C$

$B \equiv i < |ciudades|$

$I \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes$

$f_v \equiv |ciudades| - i$

Como la última instrucción del programa es el ciclo, podemos asumir que

$Q \equiv Qc \equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes$

Para demostrar la correctitud del programa basta ver que son válidas las siguientes triplas de Hoare:

$$\{Pc\}S_c\{Qc \equiv Q\} \text{ y } \{P\}S_1; S_2\{Pc\}$$

2.1.1.1 $Pc \implies wp(S_3; S_4, Qc)$ (Teorema del Invariante y Terminación)

2.1.1.1.1 $Pc \implies I$

$$\begin{aligned}
Pc \equiv res = 0 \wedge i = 0 \wedge A \wedge B \wedge C &\implies 0 \leq 0 \leq |ciudades| \wedge_L 0 = \sum_{j=0}^{-1} ciudades[j].habitantes \equiv \\
0 \leq |ciudades| \wedge_L 0 &= 0 \equiv \\
True \wedge_L True &\equiv \\
True & \\
Pc \implies I &
\end{aligned}$$

2.1.1.1.2 $I \wedge B \implies wp(S_3; S_4, I)$

$$\begin{aligned}
wp(S_3; S_4, I) &\equiv wp(res = res + ciudades[i].habitantes, wp(i = i + 1, I)) \equiv \\
wp(res = res + ciudades[i].habitantes, def(i + 1) \wedge_L I_{i+1}^i) &\equiv \\
wp(res = res + ciudades[i].habitantes, True \wedge_L 0 \leq i + 1 \leq |ciudades| \wedge_L res = \sum_{j=0}^i ciudades[j].habitantes) &\equiv \\
wp(res = res + ciudades[i].habitantes, \underbrace{-1 \leq i \leq |ciudades| - 1 \wedge_L res = \sum_{j=0}^i ciudades[j].habitantes}_{I'}) &\equiv \\
def(res + ciudades[i].habitantes) \wedge_L (I')_{res+ciudades[i].habitantes}^{res} &\equiv \\
0 \leq i < |ciudades| \wedge_L -1 \leq i \leq |ciudades| - 1 \wedge_L res + ciudades[i].habitantes = \sum_{j=0}^i ciudades[j].habitantes &\equiv \\
0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes & \\
I \wedge B \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades| &\equiv \\
0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \implies & \\
0 \leq i < |ciudades| \wedge_L \sum_{j=0}^{i-1} ciudades[j].habitantes = \sum_{j=0}^{i-1} ciudades[j].habitantes &\equiv \\
True \wedge_L True &\equiv \\
True & \\
I \wedge B \implies wp(S_3; S_4, I) &
\end{aligned}$$

2.1.1.1.3 $I \wedge \neg B \implies Qc$

$$\begin{aligned}
I \wedge \neg B \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i \geq |ciudades| &\equiv \\
i = |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes &\equiv \\
res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \implies & \\
\sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes &\equiv \\
True \wedge_L True &\equiv \\
True & \\
I \wedge \neg B \implies Qc &
\end{aligned}$$

2.1.1.1.4 $I \wedge B \wedge v_0 = f_v \implies wp(S_3; S_4, f_v < v_0)$

$$\begin{aligned}
wp(S_3; S_4, f_v < v_0) &\equiv wp(res = res + ciudades[i].habitantes, wp(i = i + 1, |ciudades| - i < v_0)) \equiv \\
wp(res = res + ciudades[i].habitantes, def(i + 1) \wedge_L (|ciudades| - i < v_0)_{i+1}^i) &\equiv \\
wp(res = res + ciudades[i].habitantes, True \wedge_L |ciudades| - i - 1 < v_0) &\equiv \\
def(res + ciudades[i].habitante) \wedge_L (|ciudades| - i - 1 < v_0)_{res+ciudades[i].habitante}^{res} &\equiv \\
0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < v_0 &\equiv
\end{aligned}$$

$$I \wedge B \wedge v_0 = f_v \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge i < |ciudades| \wedge v_0 = |ciudades| - i \equiv$$

$$\begin{aligned}
0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge v_0 = |ciudades| - i &\implies \\
0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < |ciudades| - i &\equiv \\
True \wedge_L True &\equiv \\
True & \\
I \wedge B \wedge v_0 = f_v &\implies wp(S_3; S_4, f_v < v_0)
\end{aligned}$$

2.1.1.5 $I \wedge f_v \leq 0 \implies \neg B$

$$\begin{aligned}
I \wedge f_v \leq 0 &\equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge |ciudades| - i \leq 0 \equiv \\
0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge |ciudades| \leq i &\equiv \\
i = |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes &\implies \\
i \geq |ciudades| &\equiv \\
True & \\
I \wedge f_v \leq 0 &\implies \neg B
\end{aligned}$$

2.1.2. $P \implies wp(S_1; S_2, Pc)$

$$\begin{aligned}
wp(S_1; S_2, Pc) &\equiv wp(res = 0, wp(i = 0, Pc)) \equiv wp(res = 0, def(0) \wedge_L Pc^i) \equiv \\
wp(res = 0, True \wedge_L res = 0 \wedge 0 = 0 \wedge A \wedge B \wedge C) &\equiv \\
def(0) \wedge_L (res = 0 \wedge True \wedge A \wedge B \wedge C)^{res}_0 &\equiv \\
True \wedge_L 0 = 0 \wedge A \wedge B \wedge C &\equiv True \wedge A \wedge B \wedge C \equiv A \wedge B \wedge C
\end{aligned}$$

$$P \equiv A \wedge B \wedge C \implies A \wedge B \wedge C \equiv True \wedge True \wedge True \equiv True$$

$$P \implies wp(S_1; S_2, Pc)$$

El programa es correcto con respecto a su especificación.

2.2. Demostrar que el valor devuelto es mayor a 50.000.

Informalmente, a partir de la precondition del programa se puede asumir que existe algún elemento de la secuencia de entrada con más de 50.000 habitantes. Como el ciclo, mediante el paso de las iteraciones, suma los habitantes de cada ciudad de la entrada, no hay ciudades con un numero negativo de habitantes y el ciclo finaliza, se puede concluir que la suma total de habitantes dará como resultado al menos 50.000, cumpliendo así lo pedido.

Formalmente, para demostrar que el valor devuelto es mayor a 50.000, consideramos la siguiente postcondición Q y vemos si el programa sigue siendo correcto con respecto a su especificación.

Tomando P, B, Pc, f_v igual que en el punto anterior, y

$$Q \equiv Qc \equiv res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge res > 50000$$

$$A \equiv (\exists l : \mathbb{Z})(0 \leq l < |ciudades| \wedge_L ciudades[l].habitantes > 50000)$$

$$B \equiv (\forall k : \mathbb{Z})(0 \leq k < |ciudades| \longrightarrow_L ciudades[k].habitantes \geq 0)$$

$$I \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B$$

Para demostrar la correctitud del programa basta ver que son válidas las siguientes triplas de Hoare:

$$\{Pc\} S_c \{Qc \equiv Q\} \text{ y } \{P\} S_1; S_2 \{Pc\}$$

2.2.1. $Pc \implies wp(S_3; S_4, Qc)$ (Teorema del Invariante y Terminación)

2.2.1.1 $Pc \implies I$

$$Pc \equiv res = 0 \wedge i = 0 \wedge A \wedge B \wedge C \implies 0 \leq 0 \leq |ciudades| \wedge_L 0 = \sum_{j=0}^{-1} ciudades[j].habitantes \wedge A \wedge B \equiv$$

$$\begin{aligned}
0 &\leq |ciudades| \wedge_L 0 = 0 \wedge True \wedge True \equiv \\
True &\wedge_L True \equiv \\
True & \\
Pc &\implies I
\end{aligned}$$

2.2.1.2 $I \wedge B \implies wp(S_3; S_4, I)$

$$\begin{aligned}
wp(S_3; S_4, I) &\equiv wp(res = res + ciudades.[i].habitantes, wp(i = i + 1, I)) \equiv \\
wp(res = res + ciudades.[i].habitantes, def(i + 1) \wedge_L I_{i+1}^i) &\equiv \\
wp(res = res + ciudades.[i].habitantes, True \wedge_L 0 \leq i + 1 \leq |ciudades| \wedge_L res = \sum_{j=0}^i ciudades[j].habitantes \wedge A \wedge B) &\equiv \\
wp(res = res + ciudades.[i].habitantes, \underbrace{-1 \leq i \leq |ciudades| - 1 \wedge_L res = \sum_{j=0}^i ciudades[j].habitantes \wedge A \wedge B}_{I'}) &\equiv \\
def(res + ciudades.[i].habitantes) \wedge_L (I')_{res+ciudades.[i].habitantes}^{res} &\equiv \\
0 \leq i < |ciudades| \wedge_L -1 \leq i \leq |ciudades| - 1 \wedge_L res + ciudades.[i].habitantes = \sum_{j=0}^i ciudades[j].habitantes \wedge A \wedge B &\equiv \\
0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B & \\
I \wedge B \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \wedge i < |ciudades| &\equiv \\
0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \implies & \\
0 \leq i < |ciudades| \wedge_L \sum_{j=0}^{i-1} ciudades[j].habitantes = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B &\equiv \\
True \wedge_L True \wedge True \wedge True \equiv & \\
True & \\
I \wedge B \implies wp(S_3; S_4, I) &
\end{aligned}$$

2.2.1.3 $I \wedge \neg B \implies Qc$

$$\begin{aligned}
I \wedge \neg B \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \wedge i \geq |ciudades| &\equiv \\
i = |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B &\equiv \\
res = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge A \wedge B \implies & \\
\sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes = \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes \wedge \sum_{j=0}^{|ciudades|-1} ciudades[j].habitantes > 50000 & \\
\equiv True \wedge_L True \equiv & \\
True & \\
I \wedge \neg B \implies Qc &
\end{aligned}$$

2.2.1.4 $I \wedge B \wedge v_0 = f_v \implies wp(S_3; S_4, f_v < v_0)$

$$\begin{aligned}
wp(S_3; S_4, f_v < v_0) &\equiv wp(res = res + ciudades[i].habitantes, wp(i = i + 1, |ciudades| - i < v_0)) \equiv \\
wp(res = res + ciudades[i].habitantes, def(i + 1) \wedge_L (|ciudades| - i < v_0)_{i+1}^i) &\equiv \\
wp(res = res + ciudades[i].habitantes, True \wedge_L |ciudades| - i - 1 < v_0) &\equiv \\
def(res + ciudades[i].habitante) \wedge_L (|ciudades| - i - 1 < v_0)_{res+ciudades[i].habitante}^{res} &\equiv \\
0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < v_0 &\equiv \\
I \wedge B \wedge v_0 = f_v \equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \wedge i < |ciudades| \wedge v_0 = |ciudades| - i & \\
\equiv 0 \leq i < |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \wedge v_0 = |ciudades| - i \implies & \\
0 \leq i < |ciudades| \wedge_L |ciudades| - i - 1 < |ciudades| - i &\equiv
\end{aligned}$$

$$\begin{aligned}
& True \wedge_L True \equiv \\
& True \\
& I \wedge B \wedge v_0 = f_v \implies wp(S_3; S_4, f_v < v_0)
\end{aligned}$$

2.2.1.5 $I \wedge f_v \leq 0 \implies \neg B$

$$\begin{aligned}
I \wedge f_v \leq 0 &\equiv 0 \leq i \leq |ciudades| \wedge_L res = \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \wedge |ciudades| - i \leq 0 \equiv \\
0 \leq i \leq |ciudades| \wedge_L res &= \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \wedge |ciudades| \leq i \equiv \\
i = |ciudades| \wedge_L res &= \sum_{j=0}^{i-1} ciudades[j].habitantes \wedge A \wedge B \implies \\
i \geq |ciudades| &\equiv \\
True \\
I \wedge f_v \leq 0 &\implies \neg B
\end{aligned}$$

2.2.2. $P \implies wp(S_1; S_2, Pc)$

$$\begin{aligned}
wp(S_1; S_2, Pc) &\equiv wp(res = 0, wp(i = 0, Pc)) \equiv wp(res = 0, def(0) \wedge_L Pc_0^i) \equiv \\
wp(res = 0, True \wedge_L res = 0 \wedge 0 &= 0 \wedge A \wedge B \wedge C) \equiv \\
def(0) \wedge_L (res = 0 \wedge True \wedge A \wedge B \wedge C)_0^{res} &\equiv \\
True \wedge_L 0 = 0 \wedge A \wedge B \wedge C &\equiv True \wedge A \wedge B \wedge C \equiv A \wedge B \wedge C \\
P \equiv A \wedge B \wedge C &\implies A \wedge B \wedge C \equiv True \wedge True \wedge True \equiv True
\end{aligned}$$

$$P \implies wp(S_1; S_2, Pc)$$

El programa es correcto con respecto a su especificación.