DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo Práctico 1: Especificación y WP

Elecciones Nacionales

17 de septiembre de 2023 — Algoritmos y Estructuras de Datos

Grupo.java

Integrante	LU	Correo electrónico
Pujia, Lucas	481/23	lucas.pujia@gmail.com
Praino, Luna	77/23	prainolunaa@gmail.com
Rozas, Antuanette	571/23	antuanetterozas@gmail.com
Miranda, Santiago	418/18	san_chan97@hotmail.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

1. **hayBallotage**: verifica si hay ballotage en la elección presidencial.

   proc `hayBallotage` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) : Bool

   `requiere` $\{condicionesBasicas(escrutinio, 1)\}$

   `asegura` $\{res = \neg(tieneMas45\,\%(escrutinio) \vee tieneMas40\,\%Diferencia10\,\%(escrutinio)\}$

   `pred tieneMas45%` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) {

   $(\exists i : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1 \wedge_L escrutinio[i] \neq 0 \wedge_L porcentaje(escrutinio, i) > 45)$

   }

   `pred tieneMas40%Diferencia10%` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) {

   $(\exists i : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1 \wedge_L escrutinio[i] \neq 0 \wedge_L (porcentaje(escrutinio, i) > 40 \wedge$
   $(\forall j : \mathbb{Z})\ ((0 \leq j < |escrutinio| - 1 \wedge_L escrutinio[j] \neq 0 \wedge j \neq i) \wedge_L$
   $porcentaje(escrutinio, i) - porcentaje(escrutinio, j) > 10)))$

   }

2. **hayFraude**: verifica que los votos válidos de los tres tipos de cargos electivos sumen lo mismo.

   proc `hayFraude` (in escrutinio_presidencial: $seq\langle\mathbb{Z}\rangle$, in escrutinio_senadores: $seq\langle\mathbb{Z}\rangle$, in escrutinio_diputados: $seq\langle\mathbb{Z}\rangle$) : Bool

   `requiere` $\{condicionesBasicas(escrutinio\_presidencial, 1) \wedge condicionesBasicas(escrutinio\_senadores, 2) \wedge condicionesBasicas(escrutinio\_diputados, 1) \wedge algunoSuperaUmbralElectoral(escrutinio\_diputados)\}$

   `asegura` $\{res = \text{true} \iff sumaTotal(escrutinio\_presidencial) = sumaTotal(escrutinio\_senadores) \wedge sumaTotal(escrutinio\_presidencial) = sumaTotal(escrutinio\_diputados\}$

3. **obtenerSenadoresEnProvincia**: obtiene los id de los partidos (primero y segundo) para la elección de senadores en una provincia. El id es el índice de las listas escrutinios.

   proc `obtenerSenadoresEnProvincia` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} \times \mathbb{Z}$

   `requiere` $\{condicionesBasicas(escrutinio, 2)\}$

   `asegura` $\{esElPrimerPartido(escrutinio, res_0) \wedge esElSegundoPartido(escrutinio, res_1)\}$

   `pred esElPrimerPartido` (s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {

   $i \neq |s| - 1 \wedge (\forall j : \mathbb{Z})\ (0 \leq j < |s| \longrightarrow_L s[i] \geq s[j])$

   }

   `pred esElSegundoPartido` (s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {

   $i \neq |s| - 1 \wedge \neg esElPrimerPartido(s, s[i]) \wedge (\forall j : \mathbb{Z})\ (0 \leq j < |s| \wedge \neg esElPrimerPartido(s, s[j]) \longrightarrow_L s[i] \geq s[j])$

   }

4. **calcularDHondtEnProvincia**: calcula los cocientes según el método d'Hondt para diputados en una provincia (importante: no es necesario ordenar los partidos por cantidad de votos)

   proc `calcularDHondtEnProvincia` (in cant_bancas: $\mathbb{Z}$, in escrutinio: $seq\langle\mathbb{Z}\rangle$) : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$

   `requiere` $\{condicionesBasicas(escrutinio, 1) \wedge_L algunoSuperaUmbralElectoral(escrutinio) \wedge cant\_bancas > 0\}$

   `asegura` $\{|res| = |escrutinio| - 1 \wedge_L (\forall i, j : \mathbb{Z})\ ($
   $0 \leq i < |res| \longrightarrow_L |res[i]| = cant\_bancas \wedge_L$
   $0 \leq j < |res[i]| \longrightarrow_L res[i][j] = \text{if } porcentaje(escrutinio, i) > 3 \text{ then } \lfloor \frac{escrutinio[i]}{1+j} \rfloor \text{ else } 0 \text{ fi}$
   $)\}$

5. **obtenerDiputadosEnProvincia**: calcula la cantidad de bancas de diputados obtenidas por cada partido en una provincia

   proc `obtenerDiputadosEnProvincia` (in cant_bancas: $\mathbb{Z}$, in escrutinio: $seq\langle\mathbb{Z}\rangle$, in dHondt: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) : $seq\langle\mathbb{Z}\rangle$

   `requiere` $\{esMatriz(dHondt) \wedge_L (cocientesOrdenados(dHondt) \wedge cocientesPositivos(dHondt) \wedge noExistenCocientesRepetidos(dHondt) \wedge correspondeEscrutinio(dHondt, escrutinio) \wedge 0 < cant\_bancas \leq |dHondt[0]|)\}$

asegura $\{|res| = |dHondt| \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L 0 \leq res[i] \leq cant\_bancas) \wedge \sum_{i=0}^{|res|-1} res[i] = cant\_bancas \wedge distribucionBancasCocientes(cant\_bancas, dHondt, res)\}$

pred `esDHondtValido` (in dH: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$esMatriz(dH) \wedge_L (cocientesOrdenados(dH) \wedge cocientesPositivos(dH))$

}

pred `esMatriz` (in matriz: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$|matriz| > 0 \wedge_L |matriz[0]| > 0 \wedge_L ((\forall l : \mathbb{Z})\ (0 < l < |matriz| \longrightarrow_L |matriz[l]| = |matriz[l-1]|)$

}

pred `cocientesOrdenados` (in dH: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |dH| \wedge 0 \leq j < |dH[0]| \longrightarrow_L$
$(dH[i][0] \geq dH[i][j] * (j+1) \wedge dH[i][0] < (dH[i][j]+1) * (j+1))))$

}

pred `cocientesPositivos` (in dH: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |dH| \wedge 0 \leq j < |dH[0]| \longrightarrow_L dH[i][j] \geq 0)$

}

pred `noExistenCocientesRepetidos` (in dH: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j, i', j' : \mathbb{Z})\ (0 \leq i, i' < |dH| \wedge 0 \leq j, j' < |dH[0]| \wedge (i = i' \longrightarrow j \neq j') \longrightarrow_L dH[i][j] \neq dH[i'][j'])$

}

pred `correspondeEscrutinio` (in dH: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in escr: $seq\langle \mathbb{Z}\rangle$) {

$(\forall partido : \mathbb{Z})\ (0 \leq partido < |dH| \longrightarrow_L (\forall j : \mathbb{Z})\ (0 \leq j < |dH[partido]| \longrightarrow_L$
$dH[partido][j] = \text{if } porcentaje(escrutinio, partido) > 3 \text{ then } \lfloor \frac{escrutinio[partido]}{1+j} \rfloor \text{ else } 0 \text{ fi}))$

}

pred `distribucionBancasCocientes` (in cant_bancas: $\mathbb{Z}$, in dH: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in res: $seq\langle \mathbb{Z}\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L$
$(res[i] \neq cant\_bancas \longrightarrow_L \sum_{part=0}^{|dH|-1} \sum_{coc=0}^{|dH[0]|-1} \text{if } dH[part][coc] > dH[i][res[i]] \text{ then } 1 \text{ else } 0 \text{ fi} > cant\_bancas) \wedge$
$(res[i] = cant\_bancas \longrightarrow_L (\forall j : \mathbb{Z})\ ((0 \leq j < |res| \wedge j \neq i) \longrightarrow_L dh[i][res[i]-1] > dH[j][0]))$

}

6. **validarListasDiputadosEnProvincia**: verifica que la listas de diputados de cada partido en una provincia contenga exactamente la misma cantidad de candidatos que bancas en disputa en esa provincia, y que además se cumpla la alternancia de géneros.

proc `validarListasDiputadosEnProvincia` (in cant_bancas: $\mathbb{Z}$, in listas: $seq\langle seq\langle \text{dni} : \mathbb{Z} \times \text{genero} : \mathbb{Z}\rangle\rangle$) : Bool

requiere $\{cant\_bancas > 0 \wedge generoEs1o0(listas[i]) \wedge |listas| > 0\}$

asegura $\{res = \text{true} \iff (\forall partido : \mathbb{Z})\ ($
$0 \leq partido < |listas| \longrightarrow_L |listas[partido]| = cant\_bancas \wedge estaAlternado(listas[partido])$
$)\}$

pred `generoEs1o0` (in listas: $seq\langle seq\langle \text{dni} : \mathbb{Z} \times \text{genero} : \mathbb{Z}\rangle\rangle$) {

$(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \longrightarrow_L (\forall j : \mathbb{Z})\ ($
$0 \leq j < |listas[i]| \longrightarrow_L listas[i][j]_1 = 1 \vee listas[i][j]_1 = 0$
$))$

}

pred `estaAlternado` (in lista: $seq\langle \text{dni} : \mathbb{Z} \times \text{genero} : \mathbb{Z}\rangle$) {

$(\forall j : \mathbb{Z})\ (0 < j < |lista[j]| \longrightarrow_L lista[j]_1 \neq lista[j-1]_1)$

}

7. **Auxiliares y predicados sueltos**

pred `condicionesBasicas` (in escrutinio: $seq\langle \mathbb{Z}\rangle$, in partidos : $\mathbb{Z}$) {

$hayPartidosYblancos(escrutinio, partidos) \wedge_L (\neg hayVotosNegativos(escrutinio) \wedge \neg hayEmpate(escrutinio))$

}

`pred hayPartidosYblancos` (in escrutinio: $seq\langle\mathbb{Z}\rangle$, in partidos : $\mathbb{Z}$) {

$|escrutinio| \geq partidos + 1$

}

`pred hayVotosNegativos` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) {

$(\exists i : \mathbb{Z})\ (0 \leq i < |escrutinio| \wedge_L escrutinio[i] < 0)$

}

`pred hayEmpates` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) {

$(\exists i, j : \mathbb{Z})\ ((0 \leq i, j < |escrutinio| - 1 \wedge i \neq j) \wedge_L escrutinio[i] = escrutinio[j])$

}

`pred algunoSuperaUmbralElectoral` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) {

$(\exists partido : \mathbb{Z})\ (partido \neq |escrutinio| - 1 \wedge escrutinio[partido] \neq 0 \wedge_L porcentaje(partido, escrutinio) > 3)$

}

`aux sumaTotal` (in lista: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} = \sum_{i=0}^{|lista|-1} lista[i]$;

`aux porcentaje` (in lista: $seq\langle\mathbb{Z}\rangle$, in indice: $\mathbb{Z}$) : $\mathbb{R} = 100 * \frac{sumaTotal(lista)}{lista[indice]}$;

2. Implementación y demostraciones de correctitud

1. **hayBallotage**

```

i := 0
partido1 := 0
partido2 := 0
suma := 0
k := 0

while (i < escrutinio.size()) do

    suma := suma + escrutinio[i]
    i := i + 1

endwhile

while ( k < escrutinio.size() - 1) do

    if((escrutinio[k] * 100 / suma) > partido1) then
        partido1 := escrutinio[k] * 100 / suma
    else
        if((escrutinio[k] * 100 / suma) > partido2) then
            partido2 := escrutinio[k] * 100 / suma
        else
            skip
        endif
    endif
    k := k + 1

endwhile

if (partido1 > 45 || (partido1 > 40 && partido1 - partido2 > 10)) then
    res := false
else
    res := true
endif
```

2. **hayFraude:**

$\texttt{proc Suma}$ (in lista: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$

$\texttt{requiere}$ {true}

$\texttt{asegura}$ $\{result = \sum_{i=0}^{|lista|-1} lista[i]\}$

```
PROC Suma(in lista: seq⟨ℤ⟩) : ℤ
Aux i : ℤ
Aux s : ℤ
i := 0
s := 0
while (i < lista.size()) do
    s := s + lista[i]
    i := i + 1
endwhile
result := s
Return

total_pres := Suma(escrutinio_presidencial)
total_dip := Suma(escrutinio_diputados)
total_sen := Suma(escrutinio_senadores)
res := total_pres = total_dip && total_dip = total_sen
```

Para facilitar la lectura y escritura, hacemos el reemplazo sintáctico de
"escrutinio_presidencial" por "escr_pres",
"escrutinio_diputados" por "escr_dip" y
"escrutinio_senadores" por "escr_sen"

$P \equiv \{($
$|escr\_pres| \geq 2 \wedge_L ((\forall i : \mathbb{Z})\ (0 \leq i < |escr\_pres| \wedge_L escr\_pres[i] \geq 0) \wedge$
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |escr\_pres| - 1 \longrightarrow_L escr\_pres[i] \neq escr\_pres[j])$
$) \wedge ($
$|escr\_dip| \geq 2 \wedge_L ((\forall i : \mathbb{Z})\ (0 \leq i < |escr\_dip| \wedge_L escr\_dip[i] \geq 0) \wedge$
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |escr\_dip| - 1 \longrightarrow_L escr\_dip[i] \neq escr\_dip[j])$
$) \wedge ($
$|escr\_sen| \geq 3 \wedge_L ((\forall i : \mathbb{Z})\ (0 \leq i < |escr\_sen| \wedge_L escr\_senl[i] \geq 0) \wedge$
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |escr\_sen| - 1 \longrightarrow_L escr\_sen[i] \neq escr\_sen[j])$
$)\}$

$$Q \equiv \{res = \text{true} \iff \sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

Necesitamos que $\{P\}\ S\ \{Q\}$. Como en el programa hay un llamado a procedimiento, primero vamos a demostrar su correctitud: $\{P_p\}\ S_p\ \{Q_p\}$, con $P_p \equiv \{\text{true}\}$ y $Q_p \equiv \{result = \sum_{k=0}^{|lista|-1} lista[k]\}$

Si probamos que

- $P_p \longrightarrow wp(i := 0; s := 0, P_c)$
- $P_c \longrightarrow wp(ciclo, Q_c)$
- $Q_c \longrightarrow wp(result := s, Q_p)$

Entonces probamos la tripla de Hoare de Suma como válida

Empecemos con el ciclo:
$P_c \equiv \{i = 0 \wedge s = 0\}$
$Q_c \equiv \{s = \sum_{k=0}^{|lista|-1} lista[k]\}$

$I \equiv \{0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k]\}$
$B \equiv \{i < |lista|\}$
$fv = |lista| - i$

$P_c \longrightarrow I$

$\{i = 0 \wedge s = 0\} \longrightarrow \{0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k]\}$

Como $i = 0$ entonces $i \geq 0$, y luego $s = \sum_{k=0}^{-1} lista[k]$, que es una sumatoria con un rango vacío, por lo tanto es igual a 0, y se cumple pues $s = 0$

$\{I \wedge B\}S\{I\}$

Veo si $\{I \wedge B\} \longrightarrow wp(S, I)$

$\equiv wp(s := s + lista[i]; i := i + 1, 0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k])$

$\equiv wp(s := s + lista[i], wp(i := i + 1, 0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k])$

$\equiv wp(s := s + lista[i], def(i + 1) \wedge_L 0 \leq i + 1 \leq |lista| \wedge_L s = \sum_{k=0}^{i+1-1} lista[k]$

$\equiv wp(s := s + lista[i], \text{true} \wedge_L -1 \leq i < |lista| \wedge_L s = \sum_{k=0}^{i} lista[k]$

$\equiv def(s + lista[i]) \wedge_L -1 \leq i < |lista| \wedge_L s + lista[i] = \sum_{k=0}^{i} lista[k]$

$\equiv 0 \leq i < |lista| \wedge_L -1 \leq i < |lista| \wedge_L s = \sum_{k=0}^{i}(lista[k]) - lista[i]$

$\equiv 0 \leq i < |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k]$

$\{I \wedge B\} \equiv \{0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k] \wedge i < |lista|\} \equiv \{0 \leq i < |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k]\}$ y como p $\longrightarrow$ p es tautología, puedo verificar que $\{I \wedge B\}$ S $\{I\}$

$I \wedge \neg B \longrightarrow Q_c$

$\{I \wedge B\} \equiv \{0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k] \wedge i \geq |lista|\}$

$\equiv \{i = |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k]\}$

$\equiv \{i = |lista| \wedge_L s = \sum_{k=0}^{|lista|-1} lista[k]\}$

Como $q \wedge p \longrightarrow p$ es tautología, entonces puedo verificar que $I \wedge \neg B \longrightarrow Q_c$

Luego, por el teorema del invariante, el ciclo es parcialmente correcto. Sabemos que si termina, es correcto respecto a su especificación. Ahora falta verificar que termine:

$\{I \wedge B \wedge fv = v_0\}S\{fv < v_0\}$

Veo si $\{I \wedge B \wedge fv = v_0\} \longrightarrow wp(S, fv < v_0)$

$\equiv wp(s := s + lista[i]; i := i + 1, |lista| - i < v_0)$

$\equiv wp(s := s + lista[i], wp(i := i + 1, |lista| - i < v_0))$

$\equiv wp(s := s + lista[i], def(i + 1) \wedge_L |lista| - (i + 1) < v_0)$

$\equiv wp(s := s + lista[i], \text{true} \wedge_L |lista| - i - 1 < v_0)$

$\equiv def(s + lista[i]) \wedge_L |lista| - i - 1 < v_0$

$\equiv 0 \leq i < |lista| \wedge_L |lista| - i - 1 < v_0$

$\{I \wedge B \wedge fv = v_0\} \equiv (0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k] \wedge i < |lista| \wedge fv = v_0)$

- $0 \leq i < |lista|$ es verdadero pues $0 \leq i \leq |lista|$ lo implica
- $|lista| - i - 1 < v_0$ también lo es porque $fv = |lista| - 1 = v_0$, si se reemplaza en la desigualdad anterior queda $|lista| - i - 1 < |lista| - i \iff -1 < 0$, que es verdadero

Por lo tanto, verifiqué que $\{I \wedge B \wedge fv = v_0\}$ S $\{ fv < v_0 \}$

$I \wedge fv \leq 0 \longrightarrow \neg B$

$\equiv (0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k] \wedge |lista| - i \leq 0) \longrightarrow \neg(i < |lista|)$

$\equiv (0 \leq i \leq |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k] \wedge |lista| \leq i) \longrightarrow i \geq |lista|$

$\equiv (i = |lista| \wedge_L s = \sum_{k=0}^{i-1} lista[k]) \longrightarrow i \geq |lista|$

Lo que es verdadero pues $i = |lista| \longrightarrow i \geq |lista|$. Entonces verifiqué que $I \wedge fv \leq 0 \longrightarrow \neg B$.

Por el teorema de terminación de un ciclo, probamos que el ciclo termina. Y en conjunto a lo probado anteriormente, el ciclo es correcto respecto a la especificación y termina en una cantidad finita de pasos. Ahora, siguiendo con la correctitud del procedimiento Suma, necesitamos probar que

- $P_p \longrightarrow wp(i := 0; s := 0, P_c)$
- $Q_c \longrightarrow wp(result := s, Q_p)$

$wp(i := 0; s := 0, P_c)$

$\equiv wp(i := 0; s := 0, i = 0 \wedge s = 0)$

$\equiv wp(i := 0, wp(s := 0, i = 0 \wedge s = 0)$

$\equiv wp(i := 0, def(0) \wedge_L i = 0 \wedge 0 = 0)$

$\equiv wp(i := 0, \text{true} \wedge_L i = 0)$

$\equiv def(0) \wedge_L 0 = 0) \equiv \text{true}$

$wp(result := s, Q_p)$

$\equiv wp(result := s, result = \sum_{k=0}^{|lista|-1} lista[k])$

$\equiv def(s) \wedge_L s = \sum_{k=0}^{|lista|-1} lista[k]$

$\equiv s = \sum_{k=0}^{|lista|-1} lista[k]$

Como $P_p \equiv \{\text{true}\}$ implica true y $Q_c \equiv \{s = \sum_{k=0}^{|lista|-1} lista[k]\}$ implica $s = \sum_{k=0}^{|lista|-1} lista[k]$, ambos pues $p \longrightarrow p$ es tautología, entonces probamos que la implementación de la especificación Suma es correcta.

Ahora, volviendo a la implementación de hayFraude, necesitamos probar que $P \longrightarrow ws(S, Q)$

$\{\text{true}\}$

$$\{(\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

$$\{\forall r.(r = \sum_{i=0}^{|escr\_pres|-1} escr\_pres[i]) \longrightarrow (r = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1}) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

total_pres := Suma(escrutinio_presidencial)

$$\{(total\_pres = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1}) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

$$\{\forall r.(r = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i]) \longrightarrow (total\_pres = r \wedge r = \sum_{i=0}^{|escr\_sen|-1}) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

total_dip := Suma(escrutinio_diputados)

$$\{(total\_pres = total\_dip \wedge total\_dip = \sum_{i=0}^{|escr\_sen|-1}) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \wedge \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

$$\{(\forall r.(r = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]) \longrightarrow total\_pres = total\_dip \land total\_dip = r) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \land \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

total_sen := Suma(escrutinio_senadores)

$$\{(total\_pres = total\_dip \land total\_dip = total\_sen) = \text{true} \iff$$
$$\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \land \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

res := total_pres = total_dip && total_dip = total_sen

$$\{res = \text{true} \iff \sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \land \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\}$$

Como $\{(\sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \land \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]) = \text{true} \iff \sum_{i=0}^{|escr\_pres|-1} escr\_pres[i] = \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] \land \sum_{i=0}^{|escr\_dip|-1} escr\_dip[i] = \sum_{i=0}^{|escr\_sen|-1} escr\_sen[i]\} \equiv \{\text{true}\}$, entonces $P \longrightarrow ws(S, Q)$ y por lo tanto se cumple la tripla de Hoare $\{P\}\ S\ \{Q\}$, haciendo correcto este programa respecto a la especificación de hayFraude.

3. **obtenerSenadores:**

```
i := 2
if(s[0] > s[1]) then
    primero := 0
    segundo := 1
else
    primero := 1
    segundo := 0
endif

while (i < s.size() - 1) do
    if (s[i] > s[segundo])
        if (s[i] > s[primero])
            segundo := primero
            primero := i
        else
            segundo := i
    else
        skip
    i := i + 1
endwhile
res_0 := primero
res_1 := segundo
```

Demostracion de correctitud, ejercicio 3

Planteo de Pc, Qc, B, I y fv

- $Pc = \{i = 2 \land ((primero = 0 \land segundo = 1) \lor (primero = 1 \land segundo = 0))\}$
- $Qc = \{i = |s| - 1 \land esElPrimerPartido(s, primero) \land esElSegundoPartido(s, segundo)\}$
- $B = i < |s| - 1$
- $I = \{2 \leq i \leq |s| - 1 \land_L esElPrimerPartido(subseq(s, 0, i+1), primero) \land esElSegundoPartido(subseq(s, 0, i + 1), segundo)\}$
- fv $= \{(|s| - 1) - i\}$

Primera Parte, Pre $\longrightarrow$wp(codigo previo al ciclo, Pc)

- $Pre = \{condicionesBasicas(s, 2)\}$
- $Pc = \{i = 2 \land ((primero = 0 \land segundo = 1) \lor (primero = 1 \land segundo = 0))\}$

Wp(codigo pre ciclo, Pc) $\equiv$ Wp(i:= 2, Wp(If...endif, Pc)

Llamo E1 a Wp(If...endif, Pc)

Llamo E2 a Wp(i:=2, E1)

<u>E1</u>

$Wp(If...endif, Pc) \equiv def(s[0] > s[1]) \land_L ((s[0] > s[1] \land Wp(primero := 0, Wp(segundo := 1, Pc))) \lor (s[0] \leq s[1] \land Wp(primero := 1, Wp(segundo := 0, Pc)))) \equiv$

$|s| > 1 \land_L ((s[0] > s[1] \land Wp(primero := 0, Wp(segundo := 1, Pc))) \lor (s[0] \leq s[1] \land Wp(primero := 1, Wp(segundo := 0, Pc))))$

Lo resuelvo por partes y despues lo junto

$(s[0] > s[1] \land Wp(primero := 0, Wp(segundo := 1, Pc))) \equiv$

$(s[0] > s[1] \land Wp(primero := 0, def(1) \land_L \{i = 2 \land ((primero = 0 \land 1 = 1) \lor (primero = 1 \land 1 = 0))\})) \equiv$

$(s[0] > s[1] \land Wp(primero := 0, True \land_L \{i = 2 \land ((primero = 0 \land True) \lor (primero = 1 \land False))\})) \equiv$

$(s[0] > s[1] \land Wp(primero := 0, \{i = 2 \land (primero = 0) \lor False\})) \equiv$

$(s[0] > s[1] \land def(0) \land_L \{i = 2 \land (0 = 0) \lor False\}) \equiv$

$(s[0] > s[1] \wedge True \wedge_L \{i = 2 \wedge (True) \vee False\}) \equiv$

$(s[0] > s[1] \wedge \{i = 2 \wedge True\}) \equiv$

$(s[0] > s[1] \wedge \ i = 2)$

Por ahora tengo:

$|s| > 1 \wedge_L ((s[0] > s[1] \wedge i = 2) \vee (s[0] \leq s[1] \wedge Wp(primero := 1, Wp(segundo := 0, Pc))))$

Voy con la segunda parte:

$(s[0] \leq s[1] \wedge Wp(primero := 1, Wp(segundo := 0, Pc))) \equiv$

$(s[0] \leq s[1] \wedge Wp(primero := 1, def(0) \wedge_L \{i = 2 \wedge ((primero = 0 \wedge 0 = 1) \vee (primero = 1 \wedge 0 = 0))\})) \equiv$

$(s[0] \leq s[1] \wedge Wp(primero := 1, True \wedge_L \{i = 2 \wedge ((primero = 0 \wedge False) \vee (primero = 1 \wedge True))\})) \equiv$

$(s[0] \leq s[1] \wedge Wp(primero := 1, \{i = 2 \wedge (False \vee (primero = 1))\})) \equiv$

$(s[0] \leq s[1] \wedge def(1) \wedge_L \{i = 2 \wedge (False \vee (1 = 1))\}) \equiv$

$(s[0] \leq s[1] \wedge def(1) \wedge_L \{i = 2 \wedge (False \vee True)\}) \equiv$

$(s[0] \leq s[1] \wedge True \wedge_L \{i = 2 \wedge True\}) \equiv$

$(s[0] \leq s[1] \wedge i = 2)$

Ahora tengo

$|s| > 1 \wedge_L ((s[0] > s[1] \wedge i = 2) \vee (s[0] \leq s[1] \wedge i = 2)) \equiv$ **E1**

Ahora Resuelvo **E2**

$E2 \equiv Wp(i := 2, E1) \equiv def(2) \wedge_L |s| > 1 \wedge_L ((s[0] > s[1] \wedge 2 = 2) \vee (s[0] \leq s[1] \wedge 2 = 2)) \equiv$

$True \wedge_L |s| > 1 \wedge_L ((s[0] > s[1] \wedge True) \vee (s[0] \leq s[1] \wedge True)) \equiv$

$|s| > 1 \wedge_L ((s[0] > s[1]) \vee (s[0] \leq s[1])) \equiv$ **E2**

Ahora, me queda ver si la Pre $\longrightarrow |s| > 1 \wedge_L ((s[0] > s[1]) \vee (s[0] \leq s[1]))$

- $|s| > 1$ se cumple, ya que la Pre establece que $|s| > 2$, entonces $|s| > 2 \longrightarrow |s| > 1$
- Despues, la segunda parte($(s[0] > s[1]) \vee (s[0] \leq s[1])$) siempre se va a cumplir, ya que siempre uno de los 2 casos va a ser verdadero, y entonces quedaria $True \vee False \equiv True$ o $False \vee True \equiv True$

Entonces, queda probado que Pre $\longrightarrow$Wp(codigo previo al ciclo, Pc)

Segunda Parte, Qc $\longrightarrow$wp(codigo posterior al ciclo, Post)

$Qc = \{i = |s| - 1 \wedge esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)\}$

$Post = esElPrimerPartido(s, res_0) \wedge esElSegundoPartidoPartido(s, res_1)$

Wp(codigo post ciclo, Post) $\equiv$ Wp($res_0 := primero$, Wp($res_1 := segundo$, Post))

Llamo E3 a Wp($res_1 := segundo$, Post)

Llamo E4 a Wp($res_0 := primero$, E3)

E3

Wp($res_1 := segundo$, Post)$\equiv$

$def(segundo) \wedge_L esElPrimerPartido(s, res_0) \wedge esElSegundoPartido(s, segundo) \equiv$

$True \wedge_L esElPrimerPartido(s, res_0) \wedge esElSegundoPartido(s, segundo) \equiv$

$esElPrimerPartido(s, res_0) \wedge esElSegundoPartido(s, segundo) \equiv$ **E3**

E4

Wp($res_0 := primero$, E3)$\equiv$

$def(primero) \wedge_L esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo) \equiv$

$True \wedge_L esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo) \equiv$

$esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo) \equiv$ **E4**

Ahora, tengo que ver si Qc $\longrightarrow$E4

$\{i = |s| - 1 \wedge esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)\} \longrightarrow$

$esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)$

Llamo P a $\{i = |s| - 1\}$

Llamo Q a $\{esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)\}$

Entonces, $P \wedge Q \longrightarrow Q$ es verdadero y queda probado que **Qc** $\longrightarrow$ **Wp** (codigo post ciclo, **Post**)

Tercera Parte, Pc $\longrightarrow$wp(ciclo, Qc) (con teorema del invariante)

Para probar la correcion de un ciclo tengo que:

- $Pc \longrightarrow I$
- $\{I \wedge B\}\ S\ \{I\}$
- $I \wedge \neg B \longrightarrow Qc$
- $\{I \wedge B \wedge v_0 = fv\}\ S\ \{fv < v_0\}$
- $I \wedge fv \leq 0 \longrightarrow \neg B$

<u>Pc $\longrightarrow$I</u>

$Pc = \{i = 2 \wedge ((primero = 0 \wedge segundo = 1) \vee (primero = 1 \wedge segundo = 0))\}$

$I = \{2 \leq i \leq |s| - 1 \wedge_L esElPrimerPartido(subseq(s, 0, i + 1), primero) \wedge esElSegundoPartido(subseq(s, 0, i + 1), segundo)\}$

Asumo que el antecedente es verdadero y trato de llegar al consecuente

- $i = 2 \longrightarrow 2 \leq 2 \leq |s| - 1$ Se cumple
- $(primero = 0 \wedge segundo = 1) \vee (primero = 1 \wedge segundo = 0) \longrightarrow esElPrimerPartido(subseq(s, 0, 3), primero) \wedge esElSegundoPartido(subseq(s, 0, 3), segundo)$

Llamo s_0 a la subseq(s,0,2). s_0 es la subsecuencia que unicamente toma las posiciones 0, 1 y 2 de la secuencia original s.
Entonces

$(primero = 0 \wedge segundo = 1) \vee (primero = 1 \wedge segundo = 0) \longrightarrow$
$esElPrimerPartido(s_0, primero) \wedge esElSegundoPartido(s_0, segundo)$

Por $(primero = 0 \wedge segundo = 1) \vee (primero = 1 \wedge segundo = 0)$ se que alguna de esas opciones va a ser verdadera, y en cualquier opcion de esas $primero > segundo$ siempre va a pasar. Tambien, se que la posicion 2 (los votos en blanco) no va a ser tenia en cuenta por los predicados esElPrimerPartido y esElSegundoPartido.

Entonces, $esElPrimeroPartido(s_0, primero) \wedge esElSegundoPartido(s_0, segundo)$ siempre se va a cumplir.

Conclusion, $(primero = 0 \wedge segundo = 1) \vee (primero = 1 \wedge segundo = 0) \longrightarrow esElPrimerPartido(subseq(s, 0, 3), primero) \wedge esElSegundoPartido(subseq(s, 0, 3), segundo)$ tambien se cumple, y queda probado que **Pc** $\longrightarrow$ **I**

<u>$I \wedge \neg B \longrightarrow Qc$</u>

$I = \{2 \leq i \leq |s| - 1 \wedge_L esElPrimerPartido(subseq(s, 0, i + 1), primero) \wedge esElSegundoPartido(subseq(s, 0, i + 1), segundo)\}$

$\neg B = i \geq |s| - 1$

$Qc = \{i = |s| - 1 \wedge esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)\}$

Asumo que el antecedente es verdadero y trato de llegar al consecuente

Por I y B se que $2 \leq i \leq |s|-1 \wedge i \geq |s|-1 \longrightarrow i = |s|-1$

Entonces, $\{|s|-1 = |s|-1 \wedge esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)\} \equiv$

$\{esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)\}$

Me queda probar que:

$esElPrimerPartido(subseq(s, 0, |s|-1+1), primero) \wedge esElSegundoPartido(subseq(s, 0, |s|-1+1), segundo) \longrightarrow$

$esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)$

$subseq(s, 0, |s|-1+1) \equiv subseq(s, 0, |s|)$ y esto establece la secuencia s_2 desde la posicion 0 de la secuencia s original hasta la posicion $|s|-1$ inclusive (osea, todas las posiciones de la secuencia s original).

Entonces, $s_2 \equiv s$, y se cumple que:

$esElPrimerPartido(subseq(s, 0, |s|-1+1), primero) \wedge esElSegundoPartido(subseq(s, 0, |s|-1+1), segundo) \longrightarrow$

$esElPrimerPartido(s, primero) \wedge esElSegundoPartido(s, segundo)$

$\underline{I \wedge fv \leq 0 \longrightarrow \neg B}$

$I = \{2 \leq i \leq |s|-1 \wedge_L esElPrimerPartido(subseq(s, 0, i+1), primero) \wedge esElSegundoPartido(subseq(s, 0, i+1), segundo)\}$

$fv = |s|-1-i$

$\neg B = i \geq |s|-1$

Asumo que el antecedente es verdadero y trato de llegar al consecuente.

$I \wedge fv \leq 0 \equiv$

$I \wedge |s|-1-i \leq 0 \equiv$

$I \wedge |s|-1 \leq i \equiv \neg B$

$\underline{\{I \wedge B \wedge v_0 = fv\} S \{fv < v_0\}}$

$I \wedge B \wedge v_0 = |s|-1-i \longrightarrow Wp(If...endif, Wp(i := i+1, |s|-1-i < v_0))$

$I \wedge B \wedge v_0 = |s|-1-i \longrightarrow Wp(If...endif, def(i+1) \wedge_L |s|-1-i-1 < v_0) \equiv$

$I \wedge B \wedge v_0 = |s|-1-i \longrightarrow Wp(If...endif, True \wedge_L |s|-1-i < v_0+1)$

$I \wedge B \wedge v_0 = |s|-1-i \longrightarrow Wp(If...endif, |s|-1-i < v_0+1)$

La wp(If...endif) no me cambia , la expresion, ya que no hay ningun reemplazo de variables. Entonces:

$I \wedge B \wedge v_0 = |s|-1-i \longrightarrow |s|-1-i < v_0+1$

Es verdadero, ya que $v_0 < v_0+1$ siempre

$\underline{\{I \wedge B\}\ S\ \{I\}}$

$I = \{2 \leq i \leq |s|-1 \wedge_L esElPrimerPartido(subseq(s, 0, i+1), primero) \wedge esElSegundoPartido(subseq(s, 0, i+1), segundo)\}$

$B = i < |s|-1$

Tengo que probar que $I \wedge B \longrightarrow Wp(If...endif; i := i+1, I)$

$Wp(If...endif; i := i+1, I) \equiv$

$Wp(If...endif; Wp(i := i+1, I))$

Llamo **E1** a $Wp(i := i+1, I)$ y arranco $Wp(i := i+1, I) \equiv def(i+1) \wedge_L 2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(subseq(s, 0, i+2), primero) \wedge esElSegundoPartido(subseq(s, 0, i+2), segundo) \equiv$

Llamo $s_3 a subseq(s, 0, i+2)$

$True \wedge_L 2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, segundo) \equiv$

$2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, segundo) \equiv$**E1**

$Wp(If...endif, \mathbf{E1}) \equiv$

$def(s[i] > s[segundo]) \wedge_L (s[i] > s[segundo] \wedge Wp(If...endif, E1) \vee (s[i] \leq s[segundo] \wedge Wp(Skip, E1))) \equiv$

$True \wedge_L (s[i] > s[segundo] \wedge Wp(If...endif, E1) \vee (s[i] \leq s[segundo] \wedge Wp(Skip, E1))) \equiv$

$(s[i] > s[segundo] \wedge Wp(If...endif, E1) \vee (s[i] \leq s[segundo] \wedge Wp(Skip, E1)))$

Entro en Wp(If...endif, E1) que es la mas larga

$Wp(If...endif, E1) \equiv$

$def(s[i] > s[primero]) \wedge_L (s[i] > s[primero] \wedge Wp(segundo := primero, Wp(primero := i, E1) \vee (s[i] \leq s[primero]) \wedge Wp(segundo := i, E1))) \equiv$

$True \wedge_L (s[i] > s[primero] \wedge Wp(segundo := primero, Wp(primero := i, E1) \vee (s[i] \leq s[primero]) \wedge Wp(segundo := i, E1))) \equiv$

$(s[i] > s[primero] \wedge Wp(segundo := primero, Wp(primero := i, E1) \vee (s[i] \leq s[primero]) \wedge Wp(segundo := i, E1)))$

Entro en $Wp(segundo := primero, Wp(primero := i, E1))$

$Wp(segundo := primero, Wp(primero := i, E1)) \equiv$

$Wp(segundo := primero, def(i) \wedge_L (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i)) \wedge esElSegundoPartido(s_3, segundo)) \equiv$

$Wp(segundo := primero, True \wedge_L (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i)) \wedge esElSegundoPartido(s_3, segundo)) \equiv$

$Wp(segundo := primero, (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i)) \wedge esElSegundoPartido(s_3, segundo)) \equiv$

$def(primero) \wedge_L, (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero)) \equiv$

$True \wedge_L, (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero)) \equiv$

$(2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero)) \equiv$

Ahora, me meto con $(Wp(segundo := i, E1))$

$(Wp(segundo := i, E1)) \equiv$

$def(i) \wedge_L 2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i), \equiv$

$True \wedge_L 2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i) \equiv$

$2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i)$

Ahora, tenemos que:

$(s[i] > s[primero] \wedge Wp(segundo := primero, Wp(primero := i, E1) \vee (s[i] \leq s[primero]) \wedge Wp(segundo := i, E1))) \equiv$

$(s[i] > s[primero] \wedge (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero))) \vee$
$(s[i] \leq s[primero] \wedge (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i)))$

Por ultimo, si lo enchufamos en la expresion original, tenemos que:

$(s[i] > s[segundo] \wedge Wp(If...endif, E1) \vee (s[i] \leq s[segundo] \wedge Wp(Skip, E1))) \equiv$

$(s[i] > s[segundo] \wedge (s[i] > s[primero] \wedge (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero)))) \vee$

$(s[i] \leq s[primero] \wedge (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i)))$
$\vee (s[i] \leq s[segundo] \wedge Wp(Skip, E1)) \equiv$

$s[i] > s[segundo] \wedge (s[i] > s[primero] \wedge (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero))) \vee$

$(s[i] \leq s[primero] \wedge (2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i)))$
$\vee (s[i] \leq s[segundo] \wedge 2 \leq i+1 \leq |s|-1 \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, segundo))$

Ahora, tengo que ver si $I \wedge B \longrightarrow$ toda esta expresion. Asumo que el antecendente es verdadero y trato de llegar al consecuente.

Saco el $2 \leq i+1 \leq |s|-1$ para afuera

$2 \leq i{+}1 \leq |s|{-}1 \wedge (s[i] > s[segundo] \wedge (s[i] > s[primero] \wedge_L esElPrimerPartido(s_3, i) \wedge esElSegundoPartido(s_3, primero))) \vee$

$(s[i] \leq s[primero] \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, i))$
$\vee (s[i] \leq s[segundo] \wedge_L esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, segundo))$

- Por $I \wedge B$ se que $2 \leq i \leq |s|-1 \wedge i < |s|-1$, entonces $2 \leq i+1 \leq |s|-1$ se cumple.
- A su vez, dado que $i < |s|-1$, la expresion $s_3 = subseq(s, 0, i+2)$ siempre va a estar definida, ya que i llegara a ser $|s|-2$.

Entonces, al ser una disyuncion, siempre se va a cumplir uno de los casos principales $(s[i] > s[segundo]) \vee s[i] \leq s[segundo]$, y en cualquiera de esos casos los predicados $esElPrimerPartido(s_3, primero) \wedge esElSegundoPartido(s_3, segundo)$ se van a cumplir.

Finalmente, al probar la **Primera Parte, la Segunda Parte y la Tercera parte**, por corolario de monotonia, sabemos que $Pre \longrightarrow$ wp(programa completo, Post)

6. **validarListasDiputadosEnProvincia:**

```
res := True
partido := 0
while (partido < listas.size()) do
    if (listas[partido].size() != cant_bancas) then
        res := False
    else
        i := 0
        while (i < listas[partido].size() − 1) do
            if (listas[partido][i][1] = listas[partido][i+1][1]) then
                res := False
            else
                skip
            endif
            i := i + 1
        endwhile
    endif
    partido := partido + 1
endwhile
return res
```