TP N°1: Especificación y WP

“Elecciones Nacionales”

17 de septiembre de 2023 — Algoritmos y Estructuras de Datos

SomosTodosMontiel

Integrante	LU	Correo electrónico
Borgoglio, Maximiliano	1341/21	borgogliom@gmail.com
Moran, Matias	806/19	matiec52000@hotmail.com
Recio, Santiago	1284/21	santiagorecio23@gmail.com
Solari Lenardon, Tadeo	673/23	tadebsl@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

1.1. Ejercicio 1

hayBallotage: verifica si hay ballotage en la elección presidencial.

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

Donde:

- escrutinio: es la cantidad de votos de cada partido a nivel nacional para la elección presidencial.
- devuelve verdadero sii hay ballotage en la elección presidencial.

`proc hayBallotage` (in escrutinio : $seq\langle\mathbb{Z}\rangle$) : Bool
  `requiere` $\{esEscrutinioPresidencialValido(escrutinio)\}$
  `asegura` $\{res = true \leftrightarrow \neg hayGanador(escrutinio)\}$

`pred esEscrutinioPresidencialValido` (in s: $seq\langle\mathbb{Z}\rangle$) {
  $|s| \geq 2 \land noHayEmpate(s) \land todosPositivos(s)$
}

`pred noHayEmpate` (in s: $seq\langle\mathbb{Z}\rangle$) {
  $(\forall i, j : \mathbb{Z})\ (0 \leq i, j \leq |s| - 2 \longrightarrow_L (s[i] = s[j] \leftrightarrow i = j))$
}

`pred todosPositivos` (in s: $seq\langle\mathbb{Z}\rangle$) {
  $(\forall i : \mathbb{Z})\ (0 \leq i \leq |s| - 1 \longrightarrow_L (s[i] \geq 0))$
}

`pred hayGanador` (in s: $seq\langle\mathbb{Z}\rangle$) {
  $(\exists i : \mathbb{Z})\ (0 \leq i \leq |s| - 2 \land_L esGanador(s, i))$
}

`pred esGanador` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {
  $esMaximo(s, i) \land (tieneMasDe45(s, i) \lor tieneMasDe40YSaca10Puntos(s, i))$
}

`pred esMaximo` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {
  $(\forall j : \mathbb{Z})\ ((0 \leq j \leq |s| - 2 \land j \neq i) \longrightarrow_L s[j] < s[i])$
}

`pred tieneMasDe45` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {
  $porcentajeDeVotos(s, i) > 0{,}45$
}

`pred tieneMasDe40YSaca10Puntos` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {
  $porcentajeDeVotos(s, i) > 0{,}40 \land saca10Puntos(s, i)$
}

`aux porcentajeDeVotos` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) : $\mathbb{R}$ $= \frac{s[i]}{votosTotales(s)}$;

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

`pred saca10Puntos` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {
  $(\forall j : \mathbb{Z})\ ((0 \leq j \leq |s| - 2 \land j \neq i) \longrightarrow_L porcentajeDeVotos(s, i) - porcentajeDeVotos(s, j) > 0{,}1)$
}

1.2. Ejercicio 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

Donde:

- los tres escrutinios son a nivel nacional
- devuelve verdadero sii hay al menos dos escrutinios con diferente cantidad de votos

`proc hayFraude` (in esPres : $seq\langle\mathbb{Z}\rangle$, in esSen : $seq\langle\mathbb{Z}\rangle$, in esDip : $seq\langle\mathbb{Z}\rangle$) : Bool

`requiere` $\{esEscrutinioPresidencialValido(esPres) \land$
$esEscrutinioSenadoresValido(esSen) \land$
$esEscrutinioDiputadosValido(esDip) \land$
$|esPres| = |esSen| \land |esSen| = |esDip|\}$

`asegura` $\{res = false \leftrightarrow coincidenVotos(esPres, esSen, esDip)\}$

`pred esEscrutinioSenadoresValido` (in s: $seq\langle\mathbb{Z}\rangle$) {
$|s| \geq 3 \land noHayEmpate(s) \land todosPositivos(s)$
}

`pred esEscrutinioDiputadosValido` (in s: $seq\langle\mathbb{Z}\rangle$) {
$|s| \geq 2 \land noHayEmpate(s) \land todosPositivos(s)$
}

`pred coincidenVotos` (in s1 : $seq\langle\mathbb{Z}\rangle$, in s2 : $seq\langle\mathbb{Z}\rangle$, in s3 : $seq\langle\mathbb{Z}\rangle$) {
$votosTotales(s1) = votosTotales(s2) \land votosTotales(s2) = votosTotales(s3)$
}

1.3. Ejercicio 3

obtenerSenadoresEnProvincia: obtiene los id de los partidos (primero y segundo) para la elección de senadores en una provincia

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

Donde:

- escrutinio: es la cantidad de votos de cada partido en la provincia
- devuelve una tupla que contiene el id de los dos partidos con mayor cantidad de votos.

`proc obtenerSenadoresEnProvincia` (in escrutinio : $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z} \times \mathbb{Z}$
    `requiere` $\{esEscrutinioSenadoresValido(escrutinio)\}$
    `asegura` $\{esMaximo(escrutinio, res_0) \land esSegundo(escrutinio, res_1)\}$

`pred esSegundo` (in s: $seq\langle\mathbb{Z}\rangle$, i: $\mathbb{Z}$) {
  $(\forall j : \mathbb{Z})\ ((0 \leq j \leq |s| - 2 \land j \neq i \land \neg esMaximo(s, j)) \longrightarrow_L s[j] < s[i])$
}

1.4. Ejercicio 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 : $seq\langle\mathbb{Z}\rangle$) : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$

Donde:

- cant_bancas: es la cantidad de bancas en disputa en la provincia
- escrutinio: es la cantidad de votos de cada partido en la provincia
- devuelve la matriz de dimensión # partidos × # cocientes de los cocientes del método d'Hondt

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

`requiere` $\{cantidadDeBancasValida(cant\_Bancas) \land$
$esEscrutinioDiputadosValido(escrutinio) \land$
$|escrutinio| \geq cant\_Bancas + 1 \land$
$generaDHondtValidoSinDuplicados(cant\_Bancas, escrutinio)\}$
`asegura` $\{esDHondtDe(cant\_Bancas, escrutinio, res)\}$

`pred cantidadDeBancasValida` (cantBancas: $\mathbb{Z}$) {
$cantBancas \geq 1$
}

`pred generaDHondtValidoSinDuplicados` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$) {
$(\forall i, j, k, l : \mathbb{Z})\ ((0 \leq i, k \leq |s| - 2 \land 0 \leq j, l \leq b - 1) \longrightarrow_L (\lfloor s[i]/j \rfloor = \lfloor s[k]/l \rfloor \leftrightarrow i = k \land j = l))$
}

`pred esDHondtDe` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$esMatrizDeDimension(dHondt, |s| - 1, b) \land_L todosCocientesDeDhondt(b, s, dHondt)$
}

`pred esMatrizDeDimension` (in m: $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in filas: $\mathbb{Z}$, in columnas: $\mathbb{Z}$) {
$|m| = filas \land (\forall i : \mathbb{Z})\ (0 \leq i \leq |m| - 1 \longrightarrow_L |m[i]| = columnas)$
}

`pred todosCocientesDeDhondt` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) {
$(\forall i, j : \mathbb{Z})\ ((0 \leq i \leq |s| - 2 \land 0 \leq j \leq b - 1) \longrightarrow_L (dHondt[i][j] = \lfloor s[i]/j \rfloor))$
}

1.5. Ejercicio 5

calcularDHondtEnProvincia: 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 : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) : $seq\langle\mathbb{Z}\rangle$

Donde:

- cant_bancas: es la cantidad de bancas en disputa en la provincia
- escrutinio: es la cantidad de votos de cada partido en la provincia
- dHondt: es la matriz de dimensión # partidos × # cocientes de los cocientes del método dHondt.
- devuelve la cantidad de bancas obtenidas por cada partido

`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` $\{cantidadDeBancasValida(cant\_Bancas) \wedge$
$esEscrutinioDiputadosValido(escrutinio) \wedge$
$esDHondtDe(cant\_Bancas, escrutinio, dHondt)$
$\}$

`asegura` $\{esResultadosDHondt(cant\_Bancas, listas, dHondt, res)\}$

`pred esResultadosDHondt` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in res : $seq\langle\mathbb{Z}\rangle$) {
$|res| = |s| - 1 \wedge_L sonTodosResultadosDeDHondt(b, s, dHondt, res)$
}

`pred sonTodosResultadosDeDHondt` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in res : $seq\langle\mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i \leq |res| - 1 \longrightarrow_L esCantidadDeBancasGanadas(b, s, dHondt, i, res[i]))$
}

`pred esCantidadDeBancasGanadas` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in indiceDePartido : $\mathbb{Z}$, in cantidadDeBancasGanadas : $\mathbb{Z}$) {
$cantidadDeBancasGanadas = cantidadDeVecesQuePartidoGanoUnaBanca(b, s, dHondt, indiceDePartido)$
}

`aux cantidadDeVecesQuePartidoGanoUnaBanca` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in indiceDePartido : $\mathbb{Z}$) : $\mathbb{Z} = \sum_{k=1}^{b}$ if $partidoGanoLaIesimaBanca(b, s, dHondt, indiceDePartido, k)$ then 1 else 0 fi ;

`pred partidoGanoLaIesimaBanca` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in indiceDePartido : $\mathbb{Z}$, in numeroDeBanca : $\mathbb{Z}$) {
$partidoSuperaUmbral(s, indiceDePartido) \wedge$
$partidoTieneElIesimoMaximoEnMatriz(b, s, dHondt, indiceDePartido, k)$
}

`pred partidoSuperaUmbral` (in s: $seq\langle\mathbb{Z}\rangle$, in indiceDePartido : $\mathbb{Z}$) {
$votosTotales(s, i) > 0{,}03$
}

`pred partidoTieneElIesimoMaximoEnMatriz` (in b : $\mathbb{Z}$, in s : $seq\langle\mathbb{Z}\rangle$, in dHondt : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in indiceDePartido : $\mathbb{Z}$, in iesimoMaximo : $\mathbb{Z}$) {
$(\exists j : \mathbb{Z})\ (0 \leq j \leq b - 1 \wedge_L$
$cantidadDeMayoresEnMatriz(b, s, dHondt, dHondt[indiceDePartido, j]) = iesimoMaximo - 1)$
}

`aux cantidadDeMayoresEnMatriz` (in columnas : $\mathbb{Z}$, in filas : $seq\langle\mathbb{Z}\rangle$, in matriz : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in valor : $\mathbb{Z}$) : $\mathbb{Z} = \sum_{i=0}^{filas-1} \sum_{j=0}^{columnas-1}$ if $matriz[i][j] > valor$ then 1 else 0 fi ;

1.6. Ejercicio 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 : $seq\langle seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle\rangle$) : Bool

Donde:

- cant_bancas: es la cantidad total de bancas en disputa en la provincia
- listas: son las listas de diputados de cada partido. Cada candidato/a está representado/a con una tupla que contiene el dni y el género.
- devuelve verdadero sii las listas de todos los partidos: 1) presentan la cantidad correcta de candidatos, y 2) verifican la alternancia de género.

`proc` `validarListasDiputadosEnProvincia` (in cant_Bancas : $\mathbb{Z}$, in listas : $seq\langle seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle\rangle$) : Bool
    `requiere` $\{cantidadDeBancasValida(cant\_Bancas)\}$
    `asegura` $\{res = true \leftrightarrow sonListasValidas(cant\_Bancas, listas)\}$

`pred` `sonListasValidas` (in cantBancas : $\mathbb{Z}$, listas: $seq\langle seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle\rangle$) {
    $|listas| >= 1 \land (\forall i : \mathbb{Z})\ (0 \leq i \leq |listas| - 1 \longrightarrow_L esListaAlternadaValida(cantBancas, listas[i]))$
}

`pred` `esListaAlternadaValida` (in cantBancas : $\mathbb{Z}$, lista: $seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle$) {
    $esListaValida(cantBancas, lista) \land_L esListaAlternada(lista)$
}

`pred` `esListaValida` (in cantBancas : $\mathbb{Z}$, in lista: $seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle$) {
    $|listas| = cantBancas \land (\forall i : \mathbb{Z})\ (0 \leq i \leq |lista| - 1 \longrightarrow_L (esPersonaValida(lista[i])))$
}

`pred` `esPersonaValida` (in persona : $(dni : \mathbb{Z} \times genero : \mathbb{Z})$) {
    $esDniValido(persona_0) \land esGeneroValido(persona_1)$
}

`pred` `esDniValido` (in dni : $\mathbb{Z}$) {
    $dni > 0$
}

`pred` `esGeneroValido` (in genero : $\mathbb{Z}$) {
    $genero = 1 \lor genero = 2$
}

`pred` `esListaAlternada` (lista: $seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle$) {
    $(\forall i : \mathbb{Z})\ (0 \leq i \leq |lista| - 2 \longrightarrow_L lista[i]_1 \neq lista[i+1]_1)$
}

2. Implementación

Proponer algoritmos para todos los problemas, excepto para **calcularDHondtEnProvincia** y **obtenerDiputadosEnProvincia**

2.1. candidatosMaximos

```
maximo := 0;
subMaximo := 0;
if(escrutinio[1] > escrutinio[0]) then
    maximo := 1;
else
    subMaximo := 1;
endif
i:=2;
while (i < |escrutinio| − 1) do
    if (escrutinio[i] > escrutinio[subMaximo]) then
        subMaximo := i;
    else
        skip;
    endif
    if (escrutinio[i] > escrutinio[maximo]) then
        subMaximo := maximo;
        maximo := i;
    else
        skip;
    endif
    i := i + 1;
endwhile
result := (maximo, subMaximo);
```

2.2. votosTotales

```
votosTotales := 0;
i := 0;
while (i < |escrutinio|) do
    votosTotales:= votosTotales + escrutinio[i];
    i := i + 1
endwhile
result:= votosTotales;
```

2.3. hayBallotage

```
ganador := candidatosMaximos(escrutinio)[0];
subGanador := candidatosMaximos(escrutinio)[1];
votosTotales := votosTotales(escrutinio);
porcGanador := (escrutinio[ganador] / votosTotales ) ∗ 100;
porcSubGanador := (escrutinio[subganador] / votosTotales ) ∗ 100;
if ((porcGanador > 45) || (porcGanador > 40 && ( porcGanador − porcSubganador >= 10 ))) then
    result := false
else
    result := true
endIf
```

2.4. hayfraude

```
votosPres := 0;
votosSen := 0;
votosDip := 0;
i := 0;
while(i < |escrutinio|) do
    votosPres := votosPres + escrutinio_presidencial[i];
    votosSen := votosSen + escrutinio_senadores[i];
    votosDip := votosDip + escrutinio_diputados[i];
    i := i+1;
endwhile
result := neg (votosPres = votosSen && votosSen = votosDip)
```

*aclaramos que neg es ¬

2.5. obtenerSenadoresEnProvincia

```
candidato := 0;
subCandidato := 0;
if(escrutinio[1] > escrutinio[0]) then
    candidato :=  1;
else
    subCandidato := 1;
endif
i:=2;
while (i < |escrutinio| − 1) do
    if (escrutinio[i] > escrutinio[subCandidato]) then
        subCandidato := i;
    else
        skip;
    endif
    if (escrutinio[i] > escrutinio[candidato]) then
        subCandidato := candidato;
        candidato := i;
    else
        skip;
    endif
    i := i + 1;
endwhile
result := (candidato, subCandidato);
```

2.6. validarListaDisputadosPorProvincia

```
res := true;
i := 0;
while (i < |listas|) do
    if(|listas[i]| != cant_bancas) then
        res := false;
    else
        k := 0;
        while(k < |listas[i]| − 1) do
            if (listas[i][k]_1 = listas[i][k+1]_1) then
                res:= false;
            else
                skip;
            endif
        k := k+1;
        endwhile
i := i+1;
endwhile
```

aclaramos que _ es la operacion del subindice: $listas[i][k]_1$

3. Demostración

Demostrar la correctitud de los algoritmos propuestos para los problemas **obtenerSenadoresEnProvincia** y **hayFraude** mediante el método de *weakest precondition (WP)*.

3.1. hayFraude

hayFraude tiene la siguiente especificación:

`proc hayFraude` (in esPres : $seq\langle\mathbb{Z}\rangle$, in esSen : $seq\langle\mathbb{Z}\rangle$, in esDip : $seq\langle\mathbb{Z}\rangle$) : Bool
`requiere` $\{esEscrutinioPresidencialValido(esPres) \wedge$
$esEscrutinioSenadoresValido(esSen) \wedge$
$esEscrutinioDiputadosValido(esDip) \wedge$
$|esPres| = |esSen| \wedge |esSen| = |esDip|\}$
`asegura` $\{res = false \leftrightarrow coincidenVotos(esPres, esSen, esDip)\}$

hayFraude tiene la siguiente implementación:

```
votosPres := 0;
votosSen := 0;
votosDip := 0;
i := 0;
while(i < |escrutinio|) do
    votosPres := votosPres + escrutinio_presidencial[i];
    votosSen := votosSen + escrutinio_senadores[i];
    votosDip := votosDip + escrutinio_diputados[i];
    i := i+1;
endwhile
result := neg (votosPres = votosSen && votosSen = votosDip)
```

* Aclaramos que neg es $\neg$

Por la implementación, tenemos un esquema del algoritmo de esta manera:

{Pre} S1 **{Pc}** S **{Qc}** S2 **{Post}**

Por lo que vamos a desarrollar la siguiente estrategia:

1. $Pre \implies wp(S1, Pc)$
2. {Pc} C {Qc} (Teorema del Invariante)
3. $Qc \implies wp(S2, Post)$

Por Corolario de Monotonía, nos quedaría:

$$Pre \implies wp(S1; C; S2, Post) \quad \equiv \quad \{Pc\} \quad S \quad \{Qc\}$$

Comenzamos con el item 3.

$Post \quad \equiv \quad res = false \leftrightarrow coincidenVotos(escPres, escSen, escDip)$

$\equiv \quad res = false \leftrightarrow (votosTotales(escPres) = votosTotales(escSen) \wedge votosTotales(escSen) = votosTotales(escDip))$

$\equiv \quad res = false \leftrightarrow (\sum_{j=0}^{|escPres|-1} escPres[j] = \sum_{j=0}^{|escSen|-1} escSen[j] \wedge \sum_{j=0}^{|escSen|-1} escSen[j] = \sum_{j=0}^{|escDip|-1} escDip[j])$

$wp(S2, Post) \equiv wp(res = \neg(votosPres = votosSen \wedge votosSen = votosDip), Post)$

$\equiv \quad def(\neg(votosPres = votosSen \wedge votosSen = votosDip)) \quad \wedge_L \quad Post^{res}_{\neg(votosPres=votosSen \quad \wedge \quad votosSen=votosDip)}$

$\equiv \quad True \quad \wedge_L \quad Post^{res}_{\neg(votosPres=votosSen \quad \wedge \quad votosSen=votosDip)}$

$\equiv \quad Post^{res}_{\neg(votosPres=votosSen \quad \wedge \quad votosSen=votosDip)}$

$Post \quad \equiv \quad \neg(votosPres = votosSen \wedge votosSen = votosDip) = false \leftrightarrow$
$(\sum_{j=0}^{|escPres|-1} escPres[j] = \sum_{j=0}^{|escSen|-1} escSen[j] \wedge \sum_{j=0}^{|escSen|-1} escSen[j] = \sum_{j=0}^{|escDip|-1} escDip[j])$

Ahora debemos ver si Qc $\Rightarrow$ wp(s2,Post). Para eso, pedimos que Qc sea exactamente wp(s2,Post).

Continuamos con el item 2, El ciclo.

$Pc \quad \equiv \quad votosPres = 0 \wedge votosSen = 0 \wedge votosDip = 0 \wedge i = 0 \wedge Pre$

Debido al largo de la Precondición, y sumado al hecho de que no aporta información de utilidad, decidimos dejarlo como Pre a lo largo de la demostración.

Definimos el siguiente invariante y vamos a demostrar la demostración parcial del ciclo

$Qc \quad \equiv \quad \neg(votosPres = votosSen \wedge votosSen = votosDip) = false \leftrightarrow$
$(\sum_{j=0}^{|escPres|-1} escPres[j] = \sum_{j=0}^{|escSen|-1} escSen[j] \wedge \sum_{j=0}^{|escSen|-1} escSen[j] = \sum_{j=0}^{|escDip|-1} escDip[j])$

$I \quad \equiv \quad 0 \leq i \leq |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i-1} escPres[j] \wedge votosSen = \sum_{j=0}^{i-1} escSen[j] \wedge votosDip = \sum_{j=0}^{i-1} escDip[j]$

$B \quad \equiv \quad i < |escPres|$

$Pc \implies I$
queremos ver que:
$votosPres = 0 \wedge votosSen = 0 \wedge votosDip = 0 \wedge i = 0 \wedge Pre \implies I$
$i = 0 \implies 0 \leq i \leq |escPres|$
$i = 0 \quad \wedge \quad votosPres = 0 \quad \implies \quad votosPres = \sum_{j=0}^{0-1} escPres[j] \quad \equiv \quad 0 = 0 \quad \equiv \quad True$
$i = 0 \quad \wedge \quad votosSen = 0 \quad \implies \quad votosSen = \sum_{j=0}^{0-1} escSen[j] \quad \equiv \quad 0 = 0 \quad \equiv \quad True$
$i = 0 \quad \wedge \quad votosDip = 0 \quad \implies \quad votosDip = \sum_{j=0}^{0-1} escDip[j] \quad \equiv \quad 0 = 0 \quad \equiv \quad True$

Asi comprobamos que $Pc \implies I$

$I \wedge \neg B \Rightarrow Qc$
queremos ver que:
$I \wedge \neg B \implies \neg(votosPres = votosSen \wedge votosSen = votosDip) = false \leftrightarrow$
$(\sum_{j=0}^{|escPres|-1} escPres[j] = \sum_{j=0}^{|escSen|-1} escSen[j] \wedge \sum_{j=0}^{|escSen|-1} escSen[j] = \sum_{j=0}^{|escDip|-1} escDip[j])$

Por I, tenemos que:

$I \implies votosPres = \sum_{j=0}^{i-1} escPres[j]$
$I \implies votosSen = \sum_{j=0}^{i-1} escSen[j]$
$I \implies votosDip = \sum_{j=0}^{i-1} escDip[j]$
$I \implies |escPres| = |escSen| \wedge |escSen| = |escDip|$

con lo cual podriamos reemplazar las variables votos con las sumatorias y cada limite superior de cada sumatoria con el mismo valor $|escPres|$ y nos quedaria asi:

$I \wedge \neg B \implies \neg(\sum_{j=0}^{i-1} escPres[j] = \sum_{j=0}^{i-1} escSen[j] \wedge \sum_{j=0}^{i-1} escSen[j] = \sum_{j=0}^{i-1} escDip[j]) = false \leftrightarrow$
$(\sum_{j=0}^{|escPres|-1} escPres[j] = \sum_{j=0}^{|escPres|-1} escSen[j] \wedge \sum_{j=0}^{|escPres|-1} escSen[j] = \sum_{j=0}^{|escPres|-1} escDip[j])$

Por otro lado, tenemos:

$I \implies 0 \leq i \leq |escPres|$
$\neg B \implies \neg(i < |escPres|) \equiv i \geq |escPres|$

Por lo que:

$I \wedge \neg B \implies (0 \leq i \leq |escPres|) \wedge (i \geq |escPres|) \equiv i = |escPres|$

De este modo reemplazando i con $|escPres|$, Qc quedaría:

$I \wedge \neg B \implies \neg(\sum_{j=0}^{i-1} escPres[j] = \sum_{j=0}^{i-1} escSen[j] \wedge \sum_{j=0}^{i-1} escSen[j] = \sum_{j=0}^{i-1} escDip[j]) = false \leftrightarrow$
$(\sum_{j=0}^{i-1} escPres[j] = \sum_{j=0}^{i-1} escSen[j] \wedge \sum_{j=0}^{i-1} escSen[j] = \sum_{j=0}^{i-1} escDip[j])$

Así, tendríamos algo de la forma $\neg a = false \iff a$, que siempre es verdadero, y asi comprobamos que $I \wedge \neg B \implies Qc$

$\{I \wedge B\} Sc \{I\}$
Queremos ver que:
$I \wedge B \implies wp(Sc, I) \quad \equiv wp(s6; s7; s8; s9, I) \quad \equiv \quad wp(s6, wp(s7, wp(s8, wp(s9, I))))$

Llamamos $E_1 \equiv wp(s9, I) \equiv wp(i = i+1, I)$

$\equiv def(i+1) \wedge_L I^i_{i+1}$

$\equiv True \quad \wedge_L \quad 0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen = \sum_{j=0}^{i} escSen[j] \wedge votosDip = \sum_{j=0}^{i} escDip[j]$

$\equiv 0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen = \sum_{j=0}^{i} escSen[j] \wedge votosDip = \sum_{j=0}^{i} escDip[j]$

Llamamos $E_2 \equiv wp(s8, E_1) \equiv wp(votosDip = votosDip + escDip[i], E_1)$

$\equiv def(votosDip + escDip[i]) \wedge_L (E_1)^{votosDip}_{votosDip+escDip[i]}$

$\equiv (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen = \sum_{j=0}^{i} escSen[j] \wedge votosDip = \sum_{j=0}^{i} escDip[j])^{votosDip}_{votosDip+escDip[i]}$

$\equiv (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Llamamos $E_3 \equiv wp(s7, E_2) \equiv wp(votosSen = votosSen + escSen[i], E_2)$

$\equiv def(votosSen + escSen[i]) \wedge_L (E_2)^{votosSen}_{votosSen+escSen[i]}$

$\equiv (0 \le i \le |escSen|) \quad \wedge_L \quad (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])^{votosSen}_{votosSen+escSen[i]}$

$\equiv (0 \le i \le |escSen|) \quad \wedge_L \quad (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Llamamos $E_4 \equiv wp(s6, E_3) \equiv wp(votosPres = votosPres + escPres[i], E_3)$

$\equiv def(votosPres + escPres[i]) \wedge_L (E_3)^{votosPres}_{votosPres+escPres[i]}$

$\equiv (0 \le i \le |escPres|) \quad \wedge_L \quad (0 \le i \le |escSen|) \quad \wedge_L \quad (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge votosPres = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])^{votosPres}_{votosPres+escPres[i]}$

$\equiv (0 \le i \le |escPres|) \quad \wedge_L \quad (0 \le i \le |escSen|) \quad \wedge_L \quad (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres + escPres[i] = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Ahora debemos ver si $I \wedge B \implies E_4$

Queremos ver que $I \wedge B \implies (0 \le i \le |escPres|) \quad \wedge_L \quad (0 \le i \le |escSen|) \quad \wedge_L \quad (0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres + escPres[i] = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

$I \wedge B \implies (0 \le i \le |escPres| \wedge 0 \le i \le |escSen| \wedge 0 \le i \le |escDip|) \quad \wedge_L$
$(0 \le i+1 \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres + escPres[i] = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Como habiamos visto antes I tenia la forma:
$I \quad \equiv \quad 0 \le i \le |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge\, votosPres = \sum_{j=0}^{i-1} escPres[j] \wedge votosSen = \sum_{j=0}^{i-1} escSen[j] \wedge votosDip = \sum_{j=0}^{i-1} escDip[j]$

Vemos entonces que
$I \implies (0 \leq i \leq |escPres| \wedge 0 \leq i \leq |escSen| \wedge 0 \leq i \leq |escDip|)$

Con esto E_4 nos quedaria:

$I \wedge B \implies (0 \leq i+1 \leq |escPres| \wedge |escPres| = |escSen| \wedge |escSen| = |escDip|$
$\wedge votosPres + escPres[i] = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Vemos entonces que
$I \implies (|escPres| = |escSen| \wedge |escSen| = |escDip|)$

Con esto E_4 nos quedaria:

$I \wedge B \implies (0 \leq i+1 \leq |escPres|$
$\wedge votosPres + escPres[i] = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Vemos entonces que
$I \wedge B \equiv I \wedge (i < |escPres|) \implies 0 \leq i < |escPres| \implies (0 \leq i \wedge i+1 \leq |escPres|) \implies 0 \leq i+1 \leq |escPres|$

Con esto E_4 nos quedaria:

$I \wedge B \implies (votosPres + escPres[i] = \sum_{j=0}^{i} escPres[j] \wedge votosSen + escSen[i] = \sum_{j=0}^{i} escSen[j] \wedge votosDip + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Vemos entonces que podemos reemplazar las variables votos usando su valor en el invariante
$I \implies votosPres = \sum_{j=0}^{i-1} escPres[j]$
$I \implies votosSen = \sum_{j=0}^{i-1} escSen[j]$
$I \implies votosDip = \sum_{j=0}^{i-1} escDip[j]$

Con esto E_4 nos quedaria:

$I \wedge B \implies$
$(\sum_{j=0}^{i-1} escPres[j] + escPres[i] = \sum_{j=0}^{i} escPres[j]$
$\wedge \sum_{j=0}^{i-1} escSen[j] + escSen[i] = \sum_{j=0}^{i} escSen[j]$
$\wedge \sum_{j=0}^{i-1} escDip[j] + escDip[i] = \sum_{j=0}^{i} escDip[j])$

Y asi demostramos que $I \wedge B \implies E_4$ Y con demostramos que
$I \wedge B \implies wp(Sc, I) \quad \equiv wp(s6; s7; s8; s9, I) \quad \equiv \quad wp(s6, wp(s7, wp(s8, wp(s9, I))))$
$\equiv \{I \wedge B\} Sc \{I\}$

$\{I \wedge B \ \wedge v_0 = f_v\} Sc \{f_v < v_0\}$

Llamamos $f_v \equiv |escPres| - i$

Queremos ver que:
$\{I \wedge B \ \wedge v_0 = f_v\} Sc \{f_v < v_0\}$

Para eso veamos $wp(s6; s7; s8; s9, f_v < v_0)$

Llamamos $E_1 \equiv wp(s9, fv < v_0) \equiv def(i+1) \wedge_L (f_v < v_0)_{i+1}^{i}$
$\equiv True \wedge_L (|escPres| - i < v_0)_{i+1}^{i} \equiv |escPres| - i - 1 < v_0$

Como la unica instrucción que modifica i es la s9, dejamos la wp asi y le agregamos $0 \leq i < |escPres|$

Ahora debemos ver si $I \wedge B \wedge fv = v_0 \implies wp(S, fv > v_0) \equiv |escPres| - i - 1 < v_0$

$f_v = v_0 \implies |escPres| - i = v_0 \implies v_0 - 1 < v_0$

Y asi demostramos $\{I \wedge B \ \wedge v_0 = f_v\} Sc \{f_v < v_0\}$

$(I \wedge fv \leq 0) \implies \neg B$

Veamos que:

$|esPres| - i \leq 0 \implies \neg B$
$|esPres| - i \leq 0 \implies \neg(i < |escPres|)$
$|esPres| - i \leq 0 \implies |escPres| \leq i$
$|esPres| \leq i \implies |escPres| \leq i$

Y asi demostramos $I \wedge fv \leq 0 \implies \neg B$

De este modo, demostramos la correctitud del ciclo.

Por último, seguimos con el item 1.

Recordando que :
$Pc \quad \equiv \quad votosPres = 0 \wedge votosSen = 0 \wedge votosDip = 0 \wedge i = 0 \wedge Pre$

$wp(S1, Pc) \quad \equiv \quad wp(s1, wp(s2, wp(s3, wp(s4, Pc))))$

Llamamos $E_1 \quad \equiv \quad wp(s4, Pc) \quad \equiv \quad wp(i = 0, Pc) \quad \equiv \quad votosPres = 0 \wedge votosSen = 0 \wedge votosDip = 0 \wedge 0 = 0 \wedge Pre$
$\equiv \quad votosPres = 0 \wedge votosSen = 0 \wedge votosDip = 0 \wedge Pre$

Llamamos $E_2 \quad \equiv \quad wp(s3, E_1) \quad \equiv \quad wp(votosDip = 0, E_1) \quad \equiv \quad votosPres = 0 \wedge votosSen = 0 \wedge 0 = 0 \wedge Pre$
$\equiv \quad votosPres = 0 \wedge votosSen = 0 \wedge Pre$

Llamamos $E_3 \quad \equiv \quad wp(s2, E_2) \quad \equiv \quad wp(votosSen = 0, E_2) \quad \equiv \quad votosPres = 0 \wedge 0 = 0 \wedge Pre$
$\equiv \quad votosPres = 0 \wedge Pre$

Llamamos $E_4 \quad \equiv \quad wp(s1, E_3) \quad \equiv \quad wp(votosPres = 0, E_3) \quad \equiv \quad 0 = 0 \wedge Pre$
$\equiv \quad Pre$

Luego, $wp(S1, Pc) \quad \equiv \quad E_4 \quad \equiv \quad Pre$

Finalmente, debemos ver si
$Pre \implies wp(S_1, P_c) \quad \equiv \quad Pre \implies Pre \quad \equiv \quad True$

Esto sumado al hecho que $Pc \implies wp(S_c, Q_c)$ y $Qc \implies wp(S_2, Post)$ y por el colorario de la monotonia nos da que $\{Pre\}S\{Post\}$ es una tripla valida.

3.2. obtenerSenadoresEnProvincia

obtenerSenadoresEnProvincia tiene la siguiente implementación:

```
1  candidato := 0;
2  subCandidato := 0;
3  if(escrutinio[1] > escrutinio[0]) then
4      candidato := 1;
5  else
6      subCandidato := 1;
7  endif
8  i:=2;
9  while (i < |escrutinio| − 1) do
10     if (escrutinio[i] > escrutinio[subCandidato]) then
11         subCandidato := i;
12     else
13         skip;
14     endif
15     if (escrutinio[i] > escrutinio[candidato]) then
16         subCandidato := candidato;
17         candidato := i;
18     else
19         skip;
20     endif
21     i := i + 1;
22 endwhile
23 result := (candidato, subCandidato);
```

Usaremos el termino "s"para referirnos al escrutinio con el fin de lograr mayor prolijidad.
La precondicion del programa(**P**) es *esEscrutinioSenadoresValido(s)* lo que implica que:
P≡ $\{|s| \geq 3 \wedge$ *noHayEmpate(s)* $\wedge$ *todosPositivos(s)*$\}$
La **Pc** que proponemos para el ciclo while es

Pc $\equiv \{(s[1] > s[0] \wedge candidato = 1 \wedge subCandidato = 0) \vee (s[1] < s[0] \wedge candidato = 0 \wedge subCandidato = 1) \wedge$
i = 2 $\wedge$*noHayEmpate(s)*$\}$

Ahora hay que demostrar que {**P**} S {**Pc**}

wp(s1; s2; s3; s8, Pc) $\equiv wp(s1, wp(s2, wp(s3, wp(s8, Pc))))$

Llamamos E_1 a wp(s8, Pc) $\equiv wp(i := 2, Pc) \equiv$
$(s[1] > s[0] \wedge candidato = 1 \wedge subCandidato = 0) \vee (s[1] < s[0] \wedge candidato = 0 \wedge subCandidato = 1) \wedge$
2 = 2 $\wedge$*noHayEmpate(s)*

Llamamos E_2 *a* $wp(s3, E_1) \equiv wp(if \quad s[1] > s[0] \quad then \quad candidato := 1 \quad else \quad subCandidato := 1, E_1) \equiv$

$\{((s[1] > s[0] \wedge ((s[1] > s[0] \wedge 1 = 1 \wedge subCandidato = 0) \vee (s[1] < s[0] \wedge 1 = 0 \wedge subCandidato = 1)) \wedge$
2 = 2 $\wedge$*noHayEmpate(s)))*$\vee$
$((s[1] < s[0] \wedge ((s[1] > s[0] \wedge candidato = 1 \wedge 1 = 0) \vee (s[1] < s[0] \wedge candidato = 0 \wedge 1 = 1)) \wedge$
2 = 2 $\wedge$*noHayEmpate(s)))*$\} \equiv$

$\{((s[1] > s[0] \wedge ((s[1] > s[0] \wedge True \wedge subCandidato = 0) \vee False \wedge True \wedge$ *noHayEmpate(s)))*$\vee$
$(s[1] < s[0] \wedge (False \vee (s[1] < s[0] \wedge candidato = 0 \wedge True)) \wedge$
True $\wedge$*noHayEmpate(s))*$\} \equiv$

$\{((s[1] > s[0] \wedge subCandidato = 0) \wedge$ *noHayEmpate(s))*$\vee$
$((s[1] < s[0] \wedge candidato = 0) \wedge$ *noHayEmpate(s))*$\}$

Llamamos E_3 a wp(s2, E_2) $\equiv wp((subCandidato := 0$,E$_2) \equiv$

$\{((s[1] > s[0] \wedge 0 = 0) \wedge$ *noHayEmpate(s))*$\vee$
$((s[1] < s[0] \wedge candidato = 0) \wedge$ *noHayEmpate(s))*$\}$

Llamamos E_4 a wp(s1, E_3) ≡ $wp(\{(candidato := 0,\text{E}_3\})$ ≡

$\{((s[1] > s[0]\ \wedge 0 = 0) \wedge$ *noHayEmpate(s))*$\vee$
$((s[1] < s[0]\ \wedge 0 = 0) \wedge$ *noHayEmpate(s))*$\}$ ≡

$\{((s[1] > s[0]\ \wedge True) \wedge$ *noHayEmpate(s))*$\vee$
$((s[1] < s[0]\ \wedge True) \wedge$ *noHayEmpate(s))*$\}$ ≡

$\{((s[1] > s[0])\ \wedge$*noHayEmpate(s))*$\vee$
$((s[1] < s[0])\ \wedge$*noHayEmpate(s))*$\}$ ≡

noHayEmpate(s))$\}$

Y como **P** $\implies$ *noHayEmpate(s)*,se cumple que{**P**} S {**Pc**}
Ahora proponemos una invariante **I** para el ciclo:

I $\equiv |s| - 1 \geq i \geq 2 \wedge$ *noHayEmpate(s)*$\wedge$
$(\forall j : \mathbb{Z})\ ((0 \leq j < i \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato])\quad \wedge$
$(\forall k : \mathbb{Z})\ ((0 \leq k < i \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))$

Proponemos la postcondición del ciclo(**Qc**)

Qc $\equiv esMaximo(s, candidato) \wedge esSegundo(s, subCandidato) \equiv$
$(\forall j : \mathbb{Z})\ ((0 \leq j \leq |s| - 2 \wedge j \neq candidato) \longrightarrow_L s[j] < s[candidato])\quad \wedge$
$(\forall k : \mathbb{Z})\ ((0 \leq k \leq |s| - 2 \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L s[k] < s[subCandidato])$

Teniendo ya nuestras **Pc**, **Qc** y **I** definidas, empezamos a demostrar la correctitud del ciclo.

Primero tenemos que demostrar que **Pc** $\implies$ **I**

Pc dice que *noHayEmpate(s)* lo que impica el *noHayEmpate(s)* en **I**
Pc dice que i = 2 lo que implica el lsl - 1 $\geq i \geq 2$

Si i = 2, entonces $(\forall j : \mathbb{Z})\ ((0 \leq j < i \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato])$ pasaría a ser
$(\forall j : \mathbb{Z})\ ((0 \leq j < 2 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato])$
En este caso, j solo puede valer 0 o 1, pero $j \neq candidato$, por lo que **Pc** dice que $j = subCandidato$(Ya que Pc muestra que los valores 0 y 1 son asignados al candidato y subCandidato) y en ese caso s[j] siempre es menor a s[candidato] por la definicion $(s[1] > s[0]\ \wedge candidato = 1 \wedge subCandidato = 0) \vee (s[1] < s[0] \wedge candidato = 0 \wedge subCandidato = 1)$

Si i = 2, entonces $(\forall k : \mathbb{Z})\ ((0 \leq k < i \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato])$
pasaría a ser
$(\forall j : \mathbb{Z})\ ((0 \leq k < 2 \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato])$
En este caso, k solo puede valer 0 o 1, pero como $k \neq candidato \wedge k \neq subCandidato$ y en **Pc**, los valores 0 y 1 siempre estan asignados ya sea a candidato o subCandidato, la guarda siempre da False, dando el resultado de la implicacion como True.

Por lo que **Pc** $\implies$ **I**

Ahora tenemos que demostrar que $\neg$**B** $\wedge$ **I** $\implies$ **Qc**

$\neg$**B** $\equiv i \geq |s| - 1$
i $\geq |s| - 1 \wedge |s| - 1 \geq i \geq 2 \equiv i = |s| - 1$
$\neg$**B** $\wedge$ **I** $\equiv i = |s| - 1 \wedge$ *noHayEmpate(s)*$\wedge$
$(\forall j : \mathbb{Z})\ ((0 \leq j < i \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato])\quad \wedge$
$(\forall k : \mathbb{Z})\ ((0 \leq k < i \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))$

Si i = lsl - 1 entonces $0 \leq j < i \equiv 0 \leq j < |s| - 1 \equiv 0 \leq j \leq |s| - 2$, lo mismo con $0 \leq k < i$
Por lo que quedaría que
$(\forall j : \mathbb{Z})\ ((0 \leq j \leq |s| - 2 \wedge j \neq candidato) \longrightarrow_L s[j] < s[candidato])\quad \wedge$
$(\forall k : \mathbb{Z})\ ((0 \leq k \leq |s| - 2 \wedge j \neq candidato \wedge k \neq subCandidato) \longrightarrow_L s[k] < s[subCandidato])$
Lo cual es **Qc**

Por lo que confirmamos que $\neg$**B** $\wedge$ **I** $\implies$ **Qc**

Ahora vamos a demostrar que $\{\mathbf{I} \wedge \mathbf{B}\}S\{\mathbf{I}\}$

wp(S, I) $\equiv wp(s21; s15; s10, I) \equiv wp(s21, wp(s15, wp(s10, I)))$

Llamamos Q2 $\equiv wp(s10, I) \equiv wp(i := i+1, I) \equiv lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))$

Llamamos $Q3 \equiv wp(s15, Q2) \equiv wp(if \quad s[i] > s[candidato] \quad then \quad S \quad else \quad skip, Q2) \equiv$
$(s[i] > s[candidato] \wedge wp(S, Q2)) \vee (s[i] \leq s[candidato] \wedge wp(skip, Q2)) \equiv$
$(s[i] > s[candidato] \wedge wp(S, Q2)) \vee (s[i] \leq s[candidato] \wedge Q2)$

wp(candidato := i, Q2) $\equiv lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i])) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))$

wp(subCandidato := candidato, Q3) $\equiv lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))$

Por lo que me quedaría que

$Q3 \equiv wp(if \quad s[i] > s[candidato] \quad then \quad S \quad else \quad skip, Q2) \equiv$
$(s[i] > s[candidato] \wedge |s| - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ ) \vee$
$(s[i] \leq s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))\ )$

Llamamos $Q4 \equiv wp(s10, Q3) \equiv wp(if \quad s[i] > s[subCandidato] \quad then \quad S \quad else \quad skip \quad, Q3) \equiv$
$(s[i] > s[subCandidato] \wedge wp(subCandidato := i, Q3)) \vee (s[i] \leq s[subCandidato] \wedge wp(skip, Q3)) \equiv$

$(s[i] > s[subCandidato] \wedge ((s[i] > s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ ) \vee$
$(s[i] \leq s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq candidato \wedge k \neq i) \longrightarrow_L (s[k] < s[i]))\ )))) \vee$
(s[i] $\leq s[i] \wedge ((s[i] > s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ ) \vee$
$(s[i] \leq s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge \textit{noHayEmpate(s)} \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq candidato \wedge k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))\ ))))$

Ahora habría que verificar que $\mathbf{I} \wedge \mathbf{B}$

I dice que *noHayEmpate(s)* lo que implica el *noHayEmpate(s)* en **Q4** por lo que vamos a reescribirlo:

$(s[i] > s[subCandidato] \wedge (s[i] > s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ )) \vee$
$(s[i] \leq s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq candidato \wedge k \neq i) \longrightarrow_L (s[k] < s[i]))\ )) \vee$
(s[i] $\leq s[subCandidato] \wedge ((s[i] > s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq i) \longrightarrow_L (s[j] < s[i]) \quad \wedge$
$(\forall k:\mathbb{Z})\ ((0 \leq k < i+1 \wedge k \neq i \wedge k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ ) \vee$
$(s[i] \leq s[candidato] \wedge lsl - 1 \geq i+1 \geq 2 \wedge$
$(\forall j:\mathbb{Z})\ ((0 \leq j < i+1 \wedge j \neq candidato) \longrightarrow_L (s[j] < s[candidato]) \quad \wedge$

$(\forall k : \mathbb{Z})\ ((0 \leq k < i+1 \land k \neq candidato \land k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))\ ))))$

Primero vamos a verificar la primer parte del OR:

$(s[i] > s[subCandidato] \land$
$(s[i] > s[candidato] \land lsl - 1 \geq i+1 \geq 2 \land$
$(\forall j : \mathbb{Z})\ ((0 \leq j < i+1 \land j \neq i) \longrightarrow_L (s[j] < s[i])\quad \land$
$(\forall k : \mathbb{Z})\ ((0 \leq k < i+1 \land k \neq i \land k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ )) \lor$
$(s[i] \leq s[candidato] \land lsl - 1 \geq i+1 \geq 2 \land$
$(\forall j : \mathbb{Z})\ ((0 \leq j < i+1 \land j \neq candidato) \longrightarrow_L (s[j] < s[candidato])\quad \land$
$(\forall k : \mathbb{Z})\ ((0 \leq k < i+1 \land k \neq candidato \land k \neq i) \longrightarrow_L (s[k] < s[i]))\ ))$

B $\land$**I** dice $i < |s| - 1 \land |s| - 1 \geq i \geq 2 \equiv |s| - 1 > i \geq 2$
$|s| - 1 > i \geq 2 \implies |s| - 1 \geq i+1 \geq 2$

Ahora tenemos que contemplar el caso en el que $s[i] > s[candidato]$ y en el que s[i] $\leq s[candidato]$

- $s[i] > s[candidato]$:

I me implica de forma directa $(\forall k : \mathbb{Z})\ ((0 \leq k < i \land k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ )$
Con esto se que s de todo numero entre 0 y i-1 es menor a candidato, y tambien se que s[i] $\geq s[candidato]$, por lo que cualquier s entre 0 y i-1 va a ser menor a s[i], lo que confirma $(\forall j : \mathbb{Z})\ ((0 \leq j < i) \longrightarrow_L (s[j] < s[i])$
Asi que solo me quedaría probar los casos en que j=i y k=i, pero en las guardas se especifica que aparte de estar en rango, j $\neq i$ y $k \neq i$ por lo que las guardas darían False, dando las implacaciones True.

s[i] $\leq s[candidato]$

Dada la definición de **I** si s[i] $\leq s[candidato] \land s[i] > s[subCandidato]$ solo puede ser posible si $i = candidato$

En el caso de $(\forall j : \mathbb{Z})\ ((0 \leq j < i+1 \land j \neq candidato) \longrightarrow_L (s[j] < s[candidato])$, **I** ya define todos los casos de j desde 0 hasta i-1 incluido, por lo que faltaría verificar j=i, pero al i = candidato y j $\neq$ *candiato*,
la guarda da False, haciendo que la implicaion valga True.

Para la guarda de k, si reemplazamos i por candidato, nos quedaría
$(\forall k : \mathbb{Z})\ ((0 \leq k < i+1 \land k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))$
Bajo la misma logica que antes, **I** define todos los casos de k excepto de k=i, en el cual habría una contradicción con k $\neq$ *candidato*. dando asi la guarda False y la implicación True.

Ahora verifiquemos la otra parte del OR:

(s[i] $\leq s[subCandidato] \land (s[i] > s[candidato] \land lsl - 1 \geq i+1 \geq 2 \land$
$(\forall j : \mathbb{Z})\ ((0 \leq j < i+1 \land j \neq i) \longrightarrow_L (s[j] < s[i])\quad \land$
$(\forall k : \mathbb{Z})\ ((0 \leq k < i+1 \land k \neq i \land k \neq candidato) \longrightarrow_L (s[k] < s[candidato]))\ ) \lor$
$(s[i] \leq s[candidato] \land lsl - 1 \geq i+1 \geq 2 \land$
$(\forall j : \mathbb{Z})\ ((0 \leq j < i+1 \land j \neq candidato) \longrightarrow_L (s[j] < s[candidato])\quad \land$
$(\forall k : \mathbb{Z})\ ((0 \leq k < i+1 \land k \neq candidato \land k \neq subCandidato) \longrightarrow_L (s[k] < s[subCandidato]))\ ))))$

Ya implicamos el *noHayEmpate(s)* y lsl-1 $\geq i+1 \geq 2$ anteriormente

Otra vez tenemos que contemplar el caso en el que $s[i] > s[candidato]$ y en el que s[i] $\leq s[candidato]$

- $s[i] > s[candidato]$:

I me define que $s[candidato] > s[subCandidato]$, por lo que no puede existir un i
tal que $s[i] > s[candidato] \land s[i] \leq s[subCandiato]$
Por lo que la otra parte del OR debe dar True si o si.

s[i] $\leq s[candidato]$

Dada la definición de **I** cubre los casos de j y k desde 0 hasta i-1 inclusive(ver que las definiciones de ambos para todo

son iguales en **I**), por lo que solo nos quedaría verificar los casos de j=i y k=i.
Por las guardas anteriores ya nos aseguran que s[i] $\leq candidato \wedge s[i] \leq subCandidato$
Y como se cumple que noHayEmpate(s) por **I**, las implicaciones dan True.

Con esto se termina de demostrar que **I** $\wedge$**B** $\implies Q4$ y se verifica $\{\mathbf{I} \wedge \mathbf{B}\}S\{\mathbf{I}\}$

Ahora toca demostrar que el ciclo termina, para esto proponemos una funcion variante **fv** $= |s| - i - 1$

Primero verificamos que **I** $\wedge$**fv** $\leq 0 \implies \neg$**B**

I $\wedge |s| - i - 1 \leq 0 \implies i \geq |s| - 1$
$|s|$ - i - 1 $\leq 0 \equiv |s| - 1 \leq i$
Por lo que la implicacion se cumple

Por ultimo hay que verificar que $\{\mathbf{I} \wedge \mathbf{B} \wedge v0 = |s| - i - 1\}S\{|s| - i - 1 < v0\}$

wp(i:=i+1, $|s| - i - 1 < v0$) $= |s| - i - 2 < v0$

wp(if $s[i] > s[candidato]$ Then S else skip, $|s| - i - 1 < v0$) =
$(s[i] > s[candidato] \wedge wp(S, |s| - i - 2 < v0)) \vee (\wedge s[i] \leq s[candidato] \wedge |s| - i - 2 < v0)$

wp(candidato:=i, $|s| - i - 2 < v0$) $= |s| - i - 2 < v0$
wp(subCandidato:=candidato, $|s| - i - 2 < v0$) $= |s| - i - 2 < v0$

Por lo que quedaría:
$(s[i] > s[candidato] \wedge |s| - i - 2 < v0) \vee (s[i] \leq s[candidato] \wedge |s| - i - 2 < v0) \equiv$
$(s[i] > s[candidato] \vee (s[i] \leq s[candidato])) \wedge |s| - i - 2 < v0 \equiv$
$|s| - i - 2 < v0$

wp(if $s[i] > s[subCandidato]$ Then S else skip, $|s| - i - 1 < v0$) =
$(s[i] > s[subCandidato] \wedge wp(S, |s| - i - 2 < v0)) \vee (\wedge s[i] \leq s[candidato] \wedge |s| - i - 2 < v0)$

wp(subCandidato:=i, $|s| - i - 2 < v0$) $= |s| - i - 2 < v0$

Por lo que quedaría:
$(s[i] > s[subCandidato] \wedge |s| - i - 2 < v0) \vee (s[i] \leq s[subCandidato] \wedge |s| - i - 2 < v0) \equiv$
$(s[i] > s[subCandidato] \vee (s[i] \leq s[subCandidato])) \wedge |s| - i - 2 < v0 \equiv$
$|s| - i - 2 < v0$

Si $v0 = |s| - i - 1$ entonces esto implica que $|s| - i - 2 < v0$ verificando asi la tripla $\{\mathbf{I} \wedge \mathbf{B} \wedge v0 = |s| - i - 1\}S\{|s| - i - 1 < v0\}$

Con esto ya verificamos que el ciclo es correcto y termina, dandonos
Qc $\equiv esMaximo(s, candidato) \wedge esSegundo(s, subCandidato)$
Ahora solo quedaría verificar {**Qc**} S {**Q**} y terminaríamos la verificacion del programa.

Q $\equiv esMaximo(s, res_0) \wedge esSegundo(s, res_1)$
{**Qc**} res := (candidato, subCandidato) {**Q**}

wp(res := (candidato, subCandidato), Q) $\equiv esMaximo(s, candidato) \wedge esSegundo(s, subCandidato) \equiv$ **Qc**

Con este ultimo paso completado, queda demostrado que el codigo es correcto en base a la especificación dada.