DEPARTAMENTO DE COMPUTACION
Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico 1

Especificación y WP

25 de octubre de 2023 | Algoritmos y estructuras de datos

Integrante	LU	Correo electrónico
Llop, Tobias	871/22	tobiasllop@gmail.com
Pasquet, Felipe Luc	1084/22	felipe.pasquet@gmail.com
Catarraso, Sofia	1654/21	sofiapotter07@gmail.com
Dalbene, Héctor Anselmo	923/21	anseldalbene@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

`proc hayBallotage` (in escrutinio : $seq\langle\mathbb{Z}\rangle$) : Bool
`requiere` $\{|escrutinio| > 2 \land noHayEmpate(escrutinio) \land todosPositivos(escrutinio)\}$
`asegura` $\{res = \text{true} \iff (\forall i : \mathbb{Z})\ (0 \leq i < |escrutinio| \longrightarrow_L ((porcentaje(escrutinio, i) \leq 45)$
$\land (40 < porcentaje(escrutinio, i) \leq 45 \longrightarrow_L$
$(\exists j : \mathbb{Z})\ (0 \leq j < |escrutinio| \land i \neq j \land porcentaje(escrutinio, i) - 10 < porcentaje(escrutinio, j))))\}$

`pred todosPositivos` (In s :$seq\langle\mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |s|) \longrightarrow s[i] \geq 0$
}
`pred noHayEmpate` (s: $seq\langle\mathbb{Z}\rangle$) {
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |escrutinio| - 1 \land i \neq j \longrightarrow_L s[i] \neq s[j])$
}
`aux porcentaje` (s: $seq\langle\mathbb{Z}\rangle$, i : $\mathbb{Z}$) : $\mathbb{R}$ $= \frac{s[i] * 100}{\sum_{k=0}^{|s|-1} s[k]}$;

1.2. Ejercicio 2: HayFraude

`proc hayFraude` (in escrutinioPres: $seq\langle\mathbb{Z}\rangle$, in escrutinioSen: $seq\langle\mathbb{Z}\rangle$, in escrutinioDip: $seq\langle\mathbb{Z}\rangle$) : Bool
`requiere` $\{|escrutinioPres| \geq 1 \land |escrutinioSen| \geq 1 \land |escrutinioDip| \geq 1 \land$
$todosPositivos(escrutinioPres) \land todosPositivos(escrutinioSen) \land todosPositivos(escrutinioDip)\}$
`asegura` $\{res = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]\}$

1.3. Ejercicio 3: obtenerSenadoresEnProvincia

`proc obtenerSenadoresEnProvincia` (in escrutinio: $seq\langle\mathbb{Z}\rangle$) : $\mathbb{Z}$x$\mathbb{Z}$
`requiere` $\{|escrutinio| \geq 3 \land_L noHayEmpate(escrutinio) \land todosPositivos(escrutinio)\}$
`asegura` $\{(\exists i, j : \mathbb{Z})\ (0 \leq i, j < |escrutinio| - 1 \land_L i \neq j \land sonMaximos(escrutinio, i, j))$
$res_0 = i \land\ res_1 = j\}$

`pred SonMaximos` (in escrutinio: $seq\langle\mathbb{Z}\rangle$, in i: $\mathbb{Z}$, in j: $\mathbb{Z}$) {
$(\forall h : \mathbb{Z})\ (0 \leq h < |escrutinio| - 1 \land h \neq i \land h \neq j \longrightarrow_L escrutinio[i] > escrutino[j] > escrutinio[h])$
}

1.4. Ejercicio 4: calcularDHontEnProvincia

`proc calcularDHontEnProvincia` (in cantBancas: $\mathbb{Z}$, in escrutinio: $seq\langle\mathbb{Z}\rangle$) : $seq\langle seq\langle\mathbb{Z}\rangle\rangle$
`requiere` $\{|escrutinio| > 2 \land cantBancas > 0 \land todosPositivos(escrutinio) \land$
$noHayEmpate(escrutinio) \land todosSeranDistintos(escrutinio, cantBancas)\}$
`asegura` $\{esDhontValido(res, escrutinio, cantBancas)\}$

`pred esDHontValido` (in dHont $seq\langle seq\langle\mathbb{Z}\rangle\rangle$, in escrutinio: $seq\langle\mathbb{Z}\rangle$, cantBancas:$\mathbb{Z}$) {
$|dHont| = |escrutinio| - 1 \land_L (\forall i : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1) \longrightarrow$
$(|dHont[i]| = cantBancas) \land_L (\forall j : \mathbb{Z})\ (0 \leq j < cantBancas) \longrightarrow dHont[i][j] = \frac{escrutinio[i]}{j+1}$
}
`pred todosSeranDistintos` (in s: $seq\langle\mathbb{Z}\rangle$,in c: $\mathbb{Z}$) {
$(\forall i, j : \mathbb{Z})\ (0 \leq i < |s| - 1 \land_L 1 < j \leq c \longrightarrow_L (\forall k, l : \mathbb{Z})\ (0 \leq k < |s| - 1 \land_L 1 < l \leq c\ \land_L (i \neq k \lor j \neq l) \longrightarrow_L \frac{s[i]}{j} \neq \frac{s[k]}{l}))$
}

1.5. Ejercicio 5: calcularDHontEnProvincia

`proc obtenerDiputadosEnProvincia` (in cantBancas: $\mathbb{Z}$, in escrutinio: $seq\langle\mathbb{Z}\rangle$, in dHont $seq\langle seq\langle\mathbb{Z}\rangle\rangle$) : $seq\langle\mathbb{Z}\rangle$
`requiere` $\{cantBancas > 0 \land |escrutinio| > 2 \land noHayEmpate(escrutinio), \land cocientesDistintos(dHont)$
$\land esDHontValido(dHont, escrutinio, cantbancas) \land ((\exists i : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1 \land masDel3(escrutinio, i)\}$
`asegura` $\{(\forall i : \mathbb{Z})\ (0 \leq i < |res| \longrightarrow_L$ `if` $masDel3(escrutinio, escrutinio[i]$ `then`
$res[i] = (\sum_{j=1}^{CantBancas+1}$ `if` $, sumaBanca(escrutinio[i]/j, dHont, cantBancas)$
`then1 else` $0)$ `else` $res[i] = 0)\}$

`pred sumaBanca` (in cociente: $\mathbb{Z}$, in dHont: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in CantBancas: $\mathbb{Z}$) {

$cantBancas > 0 \wedge_L |dHont[0]| > 0 \longrightarrow_L (\sum_{i=0}^{|dHont|-1} \sum_{j=0}^{|dHont[0]|-1} if\ cociente \leq dHont[i][j]\ then\ 1\ else\ 0) \leq CantBancas$

}

`pred masDel3` (s: $seq\langle \mathbb{Z}\rangle$, n: $\mathbb{Z}$) {

$porcentaje(s, n) > 3$

}

`pred cocientesDistintos` (m: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$) {

$(\forall i, j, h, k : \mathbb{Z})\ (0 \leq i \leq h < |m| \wedge 0 \leq j \leq k < m[0] \wedge i \neq h \wedge j \neq k \wedge (\exists columna : seq\langle \mathbb{Z}\rangle)\ (col(columna, m, 0))$
$\wedge (porcentaje(columna, m[i][j]) > 3 \wedge porcentaje(columna, m[h][k]) > 3) \longrightarrow m[i][j] \neq m[h][k])$

}

`pred col` (in s: $seq\langle \mathbb{Z}\rangle$, in m: $seq\langle seq\langle \mathbb{Z}\rangle\rangle$, in n: $\mathbb{Z}$) {

$|s| = |m[n]| \wedge (\forall i : \mathbb{Z})\ (0 \leq i < |m[n]|) s[i] = m[i][n]$

}

1.6. Ejercicio 6: validarListasEnProvincia

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

`requiere` $\{cantBancas > 0 \wedge |listas| > 0\}$

`asegura` $\{(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \wedge_L |listas[i]| = CantBancas \wedge cumpleAlternancia(listas[i])) \iff res = \text{true}\}$

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

$(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \longrightarrow_L (listas[i]_1 = 1 \vee listas[i]_1 = 2) \wedge listas[i]_0 > 0 \wedge$
$\neg(\exists j : \mathbb{Z})\ (0 \leq j < |listas| \wedge j \neq i \wedge listas[i]_0 = listas[j]_0))$

}

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

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

}

2. Implementaciones y demostraciones de correctitud

2.1. Ejercicio 1 - Implementación

hayBallotage ≡

```
int i := 0;
int suma := 0;
while (i < escrutinio.size()) do
    suma := suma + escrutinio[i];
    i:= i + 1;
endwhile
int j := 0;
bool res := true;
int k := 0;
float porc := 100 / suma;
while (j < escrutinio.size() − 1) do
    if  (escrutinio[j] ∗ porc) > 45 then
        res := false;
    else
        skip;
    endif
    if (escrutinio[j] ∗ porc) <= 45 and (escrutinio[j] ∗ porc) > 40 then
        while (k < escrutinio.size() −1) do
            if (escrutunio[j] ∗ porc) − 10 > (escrutinio[k] ∗ porc) then
                res := false;
            elseanc
                skip;
            endif
            k := k +1;
        endwhile
    else
        skip;
    endif
    j += j + 1;
endwhile
```

2.2. Ejercicio 2 - Implementación y WP

hayFraude ≡

```
int p := 0;
int s := 0;
int d := 0;
int votosPresidente := 0;
int votosSenador := 0;
int votosDiputado := 0;
while p < escrutinioPres.size() do
    votosPresidente := votosPresidente + escrutinioPres[p];
    p := p+1;
endwhile;
while s < escrutinioSen.size() do
    votosSenador := votosSenador + escrurinioSen[s];
    s = s+1;
endwhile;

while d < escrutinioDip.size() do
    votosDiputado := votosDiputado + escrutinioDip[d];
    d := d+1;
endwhile;

if (votosDiputado = votosSenador) and (votosDiputado = votosPresidente) then
    res := false;
else
    res := true;
endif;
```

anc

Prueba de correctitud del programa

Pre $\equiv |escrutinioPres| \geq 1 \wedge |escrutinioDip| \geq 1 \wedge |escrutinioSen| \geq 1$

$S_1 \equiv p = 0$

$S_2 \equiv i = 0$

$S_3 \equiv d = 0$

$S_4 \equiv votosPresidente = 0$

$S_5 \equiv votosSenador = 0$

$S_6 \equiv votosDiputado = 0$

$S_7 \equiv \texttt{if}\ ((votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente))\ \texttt{then}\ res = false\ \texttt{else}\ res = true$

$Post \equiv res = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$

C_1

$I_1 \equiv 0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0 \wedge votosSen = 0 \wedge votosDip = 0$

$Pc_1 \equiv p = 0 \wedge s = 0 \wedge d = 0 \wedge votosPresidente = 0 \wedge votosDiputado = 0 \wedge votosSenador = 0 \wedge |escrutinioPresDipSen| > 0$

$Qc_1 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$

$B_1 \equiv p < |escrutinioPres|$

$fv_1 \equiv |escrutinioPres| - p$

$* \ Pre \longrightarrow wp(s_1, s_2, s_3, s_4, s_5, s_6, Pc_1)$

$$\begin{aligned}
wp(s_1, s_2, s_3, s_4, s_5, s_6, Pc_1) \equiv\ & def(p = 0) \wedge def(i = 0) \wedge def(d = 0) \wedge def(votosPresidente = 0) \wedge \\
& def(votosSenador = 0) \wedge def(votosDiputado = 0) \wedge (0 = 0 \wedge 0 = 0 \wedge 0 = 0 \wedge 0 = 0 \wedge 0 = 0 \wedge \\
& 0 = 0 \wedge |escrutinioPres| \geq 1 \wedge |escrutinioSen| \geq 1 \wedge |escrutinioDip| \geq 1) \\
\equiv\ & true \ \wedge true \ \wedge true \ \wedge true \ \wedge true \ \wedge true \ \wedge (true \ \wedge true \ \wedge true \ \wedge true \ \wedge true \ \wedge true \ \wedge \\
& |escrutinioPres| \geq 1 \wedge |escrutinioSen| \geq 1 \wedge |escrutinioDip| \geq 1) \\
\equiv\ & |escrutinioPres| \geq 1 \wedge |escrutinioSen| \geq 1 \wedge |escrutinioDip| \geq 1
\end{aligned}$$

$Como\ Pre = wp(s_1, s_2, s_3, s_4, s_5, s_6, Pc_1)$:

$Pre \longrightarrow wp(s_1, s_2, s_3, s_4, s_5, s_6, Pc_1) = true$

Ahora probaremos la correctitud del primer ciclo al que llamaremos C_1, utilizando el teorema del invariante

$* \, Pc_1 \longrightarrow I_1$:

$p = 0 \longrightarrow 0 \leq p \leq |escrutinioPres| = true$
$p = 0 \longrightarrow 0 \leq |escrutinioPres| = true$ Por definición de la operación longitud de secuencias
$votosPresidente = 0 \longrightarrow votosPresidente = \sum_{k=o}^{p-1} escrutinioPres[k] = true$ Porque la sumatoria es vacía y suma 0

$* \, I_1 \wedge \neg B_1 \longrightarrow Qc_1$:

$\neg B_1 \equiv p \geq |escrutinioPres|$
$I_1 \equiv 0 \leq p \leq |escrutinioPres| \longrightarrow p = |escrutinioPres| = true$
$s = 0 \wedge d = 0 = true$ por I_1
$p = |escrutinioPres| - 1 \longrightarrow votosPresidente = \sum_{k=o}^{|escrutioPres|-1} escrutinioPres[k] = true$

$* \, I_1 \wedge fv_1 \leq 0 \longrightarrow \neg B_1$:

$I_1 \equiv 0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0 \wedge votosSen = 0 \wedge votosDip = 0$

$fv_1 \equiv |escrutinioPres| - p \leq 0 \longrightarrow |escrutinioPres| = p \longrightarrow |escrutinioPres| \leq p \equiv true$

$* \, \{I_1 \wedge B_1\} C_1 \{I_1\}$:

Queremos probar $(I_1 \wedge B_1) \longrightarrow wp(C_1, I_1)$
busco $wp(votosPresidente := votosPresidente + escrPres[p]; p := p + 1, I) \equiv$
$wp(votosPres := votosPres + escrPres[p], wp(p := p + 1, I))$

$wp(p := p+1, I) \equiv def(p+1) \wedge_L I^p_{p+1} \equiv E \equiv 0 \leq p+1 \leq |escrutinioPres| \wedge_L votosPresidente = \sum_{k=o}^{p} escrutinioPres[k] \wedge$
$s = 0 \wedge d = 0$
$\equiv wp(votosPres := votosPres + escrPres[p], E) \equiv def(votosPres + escrPres[p]) \wedge_L E^{votosPres}_{votosPres+escrPres[p]} \equiv$
$\equiv 0 \leq p < |escrutinioPres| \wedge_L (-1 \leq p < |escrutinioPres| \wedge votosPres = \sum_{k=o}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0)$

Por lo tanto:
$wp(C_1, I_1) \equiv 0 \leq p < |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0)$

Ahora veamos:
$I_1 \wedge B_1 \equiv (0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0) \wedge p < |escrutinioPres|$
$\equiv 0 \leq p < |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0$

Como $(\mathsf{I}_1 \wedge \; B_1) = wp(C_1, I_1)$, queda demostrado que: $I_1 \wedge B_1 \longrightarrow wp(C_1, I_1)$

I $\wedge * \{I_1 \wedge B_1 \wedge v_0 = |escrutinioPres| - p\} \, C_1 \, \{|escrutinioPres| - p < v_0\}$
Queremos probar $(I_1 \wedge B_1 \wedge v_0 = |escrutinioPres| - p) \longrightarrow wp(C_1, |escrutinioPres| - p < v_0)$
busco $wp(votosPresidente := votosPresidente + escrPres[p]; p := p + 1, |escrutinioPres| - p < v_0) \equiv$
$wp(votosPres := votosPres + escrPres[p], wp(p := p + 1, |escrutinioPres| - p < v_0))$

$wp(p := p + 1, |escrutinioPres| - p < v_0) \equiv Q \equiv -1 \leq p < |escrutinioPres| \wedge_L |escrutinioPres| - p < v_0 + 1$

$wp(votosPres := votosPres + escrutinioPres[p], Q) \equiv 0 \leq p < |escrutinioPres| \wedge_L (-1 \leq p < |escrutinioPres| \wedge_L$
$|escrutinioPres| - p < v_0 + 1)$

$wp(C_1, |escrutinioPres| - p < v_0) \equiv 0 \leq p < |escrutinioPres| \wedge_L |escrutinioPres| < p + v_0 + 1$

Ahora veamos:
$I_1 \wedge B_1 \wedge v_0 = (|escrutinioPres| - p) \equiv (0 \leq p < |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge$
$s = 0 \wedge d = 0) \wedge |escrutinioPres| = v_0 + p$

$(I_1 \wedge B_1 \wedge v_0 = |escrutinioPres| - p) \longrightarrow wp(C_1, |escrutinioPres| - p < v_0)$?
$0 \leq p < |escrutinioPres| \longrightarrow 0 \leq p < |escrutinioPres| = true$
$|escrutinioPres| = v_0 + p \longrightarrow |escrutinioPres| < p + v_0 + 1 \longrightarrow v_0 + p < v_0 + p + 1 = true$

Queda demostrado: $\{I_1 \wedge B_1 \wedge v_0 = |escrutinioPres| - p\} \, C_1 \, \{|escrutinioPres| - p < v_0\}$

Luego, probaremos que: $Qc_1 \longrightarrow Pc_2$

$Pc_2 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k] \wedge votosDiputado = 0 \wedge votosSenador = 0 \wedge |escrutinioPres| > 0 \wedge |escrutinioSen| > 0 \wedge |escrutinioDIp| > 0$

$Qc_1 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$

Prueba
$p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \longrightarrow$
$p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$
$\equiv true$ por equivalencia

Ahora probaremos la correctitud del segundo ciclo al que llamaremos C_2, utilizando el teorema del invariante

$I_2 \equiv 0 \leq s \leq |escrutinioSen| \wedge votosSenador = \sum_{j=0}^{s-1} escrutinioSen[j] \wedge p = |escrutinioPres| \wedge d = 0 \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k] \wedge votosDip = 0$

$Pc_2 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k] \wedge votosDiputado = 0 \wedge votosSenador = 0 \wedge |escrutinioPres| > 0 \wedge |escrutinioSen| > 0 \wedge |escrutinioDIp| > 0$

$Qc_2 \equiv p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{s-1} escrutinioSen[j]$

$B_2 \equiv s < |escrutinioSen|$

$fv_2 \equiv |escrutinioSen| - s$

$* \; Pc_2 \longrightarrow I_2 :$

$s = 0 \longrightarrow 0 \leq s \leq |escrutinioSen| \equiv true$
$s = 0 \longrightarrow 0 \leq |escrutinioPres| \equiv$ Por definición de la operación longitud de secuencias
$votosSenador = 0 \longrightarrow votosSenador = \sum_{j=o}^{s-1} escrutinioSen[j] \equiv true$ Porque la sumatoria es vacía y suma 0

* $I_2 \wedge \neg B_2 \longrightarrow Qc_2 :$

$\neg B_2 \equiv s \geq |escrutinioSen|$
$I_2 \equiv 0 \leq s \leq |escrutinioSen| \longrightarrow s = |escrutinioSen| \equiv true$
$p = |escrutinioPres| \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \equiv true$ por I_2
$s = |escrutinioSen| - 1 \longrightarrow votosSenador = \sum_{j=o}^{|escrutioSen|-1} escrutinioSen[j] \equiv true$

* $I_2 \wedge fv_2 \leq 0 \longrightarrow \neg B_2 :$

$fv_2 \equiv |escrutinioSen| - s \leq 0 \longrightarrow |escrutinioSen| = s \longrightarrow |escrutinioSen| \leq s \equiv true$

*$\{I_2 \wedge B_2\} C_2 \{I_2\} :$

Queremos probar $(I_2 \wedge B_2) \longrightarrow wp(C_2, I_2)$
busco $wp(votosSenador := votosSenador + escrutinioSen[s]; s := s + 1, I_2) \equiv$
$wp(votosSenador := votosSenador + escrutinioSen[s], wp(s := s + 1, I_2))$

$wp(s := s+1, I_2) \equiv def(s+1) \wedge_L I^s_{s+1} \equiv E \equiv 0 \leq s+1 \leq |escrutinioSen| \wedge_L votosSenador = \sum_{j=o}^{s} escrutinioSen[j] \wedge$
$p = |escrutinioPres| \wedge d = 0$
$\equiv wp(votosSenador := votosSenador + escrutinioSen[s], E) \equiv def(votosSenador + escrutinioSen[s]) \wedge_L E^{votosSenador}_{votosSenador+escrutinioSen[}$

$\equiv 0 \leq s < |escrutinioSen| \wedge_L (-1 \leq s < |escrutinioSen| \wedge votosSenador = \sum_{j=o}^{s-1} escrutinioSen[j] \wedge p = |escrutinioPres| \wedge d = 0)$

Por lo tanto:
$wp(C_1, I_1) \equiv 0 \leq s < |escrutinioSen| \wedge votosSenador = \sum_{j=0}^{s-1} escrutinioSen[j] \wedge p = |escrutinioPres| \wedge d = 0)$

Ahora veamos:
$I_2 \wedge B_2 \equiv 0 \leq s < |escrutinioSen| \wedge votosSenador = \sum_{j=0}^{s-1} escrutinioSen[j] \wedge p = |escrutinioPres| \wedge d = 0$

Como ($\mathsf{I}_2 \wedge\ B_2) = wp(C_2, I_2),$ queda demostrado que: $I_2 \wedge B_2 \longrightarrow wp(C_2, I_2)$

*$\{\mathsf{I}_2 \wedge B_2 \wedge v_0 = |escrutinioSen| - s\}\, C_2\, \{|escrutinioSen| - s < v_0\}$

Queremos probar $(I_2 \wedge B_2 \wedge v_0 = |escrutinioSen| - S) \longrightarrow wp(C_2, |escrutinioSen| - S < v_0)$
busco $wp(votosSenador := votosSenador + escrutinioSen[s]; s := s+1, |escrutinioSen| - s < v_0) \equiv$
$wp(votosSenador := votosSenador + escrutinioSen[p], wp(s := s+1, |escrutinioSen| - s < v_0))$

$wp(s := s+1, |escrutinioSen| - s < v_0) \equiv Q \equiv -1 \leq s < |escrutinioSen| \wedge_L |escrutinioSen| - s < v_0 + 1$

$wp(votosSenador := votosSenador + escrutinioSenador[s], Q) \equiv 0 \leq s < |escrutinioSen| \wedge_L (-1 \leq s < |escrutinioSen| \wedge_L$
$|escrutinioSen| - s < v_0 + 1)$

$wp(C_2, |escrutinioSen| - s < v_0) \equiv 0 \leq s < |escrutinioSen| \wedge_L |escrutinioSen| < s + v_0 + 1$

Ahora veamos:
$I_2 \wedge B_2 \wedge v_0 = (|escrutinioSen| - s) \equiv (0 \leq s < |escrutinioSen| \wedge votosSenador = \sum_{j=0}^{s-1} escrutinioSen[j] \wedge$
$p = |escrutinioPres| \wedge d = 0 \wedge votosPres = \sum_{k=0}^{s-1} escrutinioPres[k]) \wedge votosDip = 0 \wedge |escrutinioSen| =$
$v_0 + s$

$(I_2 \wedge B_2 \wedge v_0 = |escrutinioSen| - s) \longrightarrow wp(C_2, |escrutinioSen| - s < v_0)$?
$0 \leq s < |escrutinioSen| \longrightarrow 0 \leq s < |escrutinioSen| \equiv true$
$|escrutinioSen| = v_0 + s \longrightarrow |escrutinioSen| < s + v_0 + 1 \longrightarrow v_0 + s < v_0 + s + 1 \equiv true$

Queda demostrado: $\{I_2 \wedge B_2 \wedge v_0 = |escrutinioSen| - s\}\, C_2\, \{|escrutinioSen| - s < v_0\}$

Luego, probaremos que: $\mathsf{Qc}_2 \longrightarrow Pc_3$

$Pc_3 \equiv Pc_3 \equiv p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge d = 0 \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k] \wedge votosDiputado = 0 \wedge votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] \wedge |escrutinioPres| > 0 \wedge |escrutinioSen| > 0 \wedge |escrutinioDip| > 0$

$Qc_2 \equiv p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{s-1} escrutinioSen[j]$

Prueba:
$p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{|escrutioSen|-1} escrutinioSen[j] \longrightarrow p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{|escrutioSen|-1} escrutinioSen[j]$

$\equiv true$ por equivalencia

Ahora probaremos la correctitud del tercer ciclo al que llamaremos $C_3,$ utilizando el teorema del invariante.

$I_3 \equiv 0 \leq d \leq |escrutinioDip| \wedge votosDiputado = \sum_{i=0}^{d-1} escrutinioDip[i] \wedge p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j]$

$Pc_3 \equiv p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge d = 0 \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k] \wedge$
$votosDiputado = 0 \wedge votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] \wedge |escrutinioPres| > 0 \wedge$
$|escrutinioSen| > 0 \wedge |escrutinioDip| > 0$

$Qc_3 \equiv |escrutinioPres| > 0 \wedge |escrutinioSen| > 0 \wedge |escrutinioDip| > 0 \wedge p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge$

$d = |escrutinioDip| \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j]$
$\wedge\ votosDiputado = \sum_{i=0}^{d-1} escrutinioSen[i]$
$B_3 \equiv d < |escrutinioDip|$

$fv_3 \equiv |escrutinioDip| - d$

$*\ Pc_3 \longrightarrow I_3:$

$d = 0 \longrightarrow 0 \leq d \leq |escrutinioDip| \equiv true$
$d = 0 \longrightarrow 0 \leq |escrutinioPres| \equiv$ Por definición de la operación longitud de secuencias
$votosDiputado = 0 \longrightarrow votosDiputado = \sum_{i=o}^{d-1} escrutinioDip[i] \equiv true$ Porque la sumatoria es vacía y suma 0

$p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k]$
$\wedge\ votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] \longrightarrow I_3 \equiv true$ Por equivalencia

* $\mathrm{I}_3 \wedge \neg B_3 \longrightarrow Qc_3:$

$\neg B_3 \equiv d \geq |escrutinioDip|$
$I_3 \equiv 0 \leq d \leq |escrutinioDip| \longrightarrow d = |escrutinioDip| \equiv true$
$p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge$
$\sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] \equiv true$ por I_3
$d = |escrutinioDip| - 1 \longrightarrow votosDiputado = \sum_{i=o}^{|escrutioDip|-1} escrutinioSen[i] \equiv true$

* $\mathrm{I}_3 \wedge fv_3 \leq 0 \longrightarrow \neg B_3:$

$fv_3 \equiv |escrutinioDip| - d \leq 0 \longrightarrow |escrutinioDip| = d \longrightarrow |escrutinioDip| \leq d \equiv true$

$*\ \{I_3 \wedge B_3\}C_3\{I_3\}:$

Queremos probar $(I_3 \wedge B_3) \longrightarrow wp(C_3, I_3)$
busco $wp(votosDiputado := votosDiputado + escrutinioDip[d], wp(d := d+1, I_3))$

$wp(d := d+1, I_3) \equiv def(d+1) \wedge_L I^d_{d+1} \equiv E \equiv 0 \leq d+1 \leq |escrutinioDip| \wedge_L votosDiputado = \sum_{i=o}^{d} escrutinioDip[i]\ \wedge$
$p = |escrutinioPres| \wedge s = |escrutinioSen|$
$\equiv wp(votosDiputado := votosDiputado{+}escrutinioDip[d], E) \equiv def(votosDiputado{+}escrutinioDip[d]) \wedge_L E^{votosDiputado}_{votosDiputado+escrutinio}$

$\equiv 0 \leq d < |escrutinioDip| \wedge_L (-1 \leq d < |escrutinioDip| \wedge votosDiputado = \sum_{i=o}^{d-1} escrutinioDip[i] \wedge$
$p = |escrutinioPres| \wedge s = |escrutinioSen|)$

Por lo tanto:
$wp(C_1, I_3) \equiv 0 \leq d < |escrutinioDip| \wedge votosDiputado = \sum_{i=0}^{d-1} escrutinioDip[i] \wedge p = |escrutinioPres|$
$\wedge\ s = |escrutinioSen|)$

Ahora veamos:
$I_3 \wedge B_3 \equiv 0 \leq d < |escrutinioDip| \wedge votosDiputado = \sum_{ji=0}^{d-1} escrutinioDip[i] \wedge p = |escrutinioPres| \wedge s = |escrutinioSen|$

Como $(I_3 \wedge\ B_3) = wp(C_3, I_3),$ queda demostrado que: $I_3 \wedge B_3 \longrightarrow wp(C_3, I_3)$

*$\{\mathrm{I}_3 \wedge B_3 \wedge v_0 = |escrutinioDip| - d\}\, C_3\, \{|escrutinioDip| - d < v_0\}$

Queremos probar $(I_3 \wedge B_3 \wedge v_0 = |escrutinioDip| - d) \longrightarrow wp(C_3, |escrutinioDip| - d < v_0)$
busco $wp(votosDiputado := votosDiputado + escrutinioDip[d]; d := d+1, |escrutinioDip| - d < v_0) \equiv$

$wp(d := d+1, |escrutinioDip| - d < v_0) \equiv Q \equiv -1 \leq d < |escrutinioDip| \wedge_L |escrutinioDip| - d < v_0 + 1$

$wp(votosDiputado := votosDiputado{+}escrutinioDiputado[d], Q) \equiv 0 \leq d < |escrutinioDip| \wedge_L (-1 \leq d < |escrutinioDip| \wedge_L$
$|escrutinioDip| - d < v_0 + 1)$

$wp(C_2, |escrutinioDip| - d < v_0) \equiv 0 \leq d < |escrutinioDip| \wedge_L |escrutinioDip| < d + v_0 + 1$

Ahora veamos:
$I_3 \wedge B_3 \wedge v_0 = (|escrutinioDip| - d) \equiv (0 \leq d < |escrutinioDip| \wedge votosDiputado = \sum_{i=0}^{d-1} escrutinioSen[i] \wedge$
$p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge votosPres = \sum_{k=0}^{|escrutinioPres|-1} escrutinioPres[k]) \wedge$
$votosSen = \sum_{j=0}^{|escrutinioSen|-1} escrutinioPres[j] \wedge |escrutinioDip| = v_0 + s$

$(I_3 \wedge B_3 \wedge v_0 = |escrutinioDip| - d) \longrightarrow wp(C_3, |escrutinioDip| - d < v_0)$?

$0 \leq d < |escrutinioDip| \longrightarrow 0 \leq d < |escrutinioDip| \equiv true$
$|escrutinioDip| = v_0 + d \longrightarrow |escrutinioDip| < d + v_0 + 1 \longrightarrow v_0 + d < v_0 + d + 1 \equiv true$

Queda demostrado: $\{I_3 \wedge B_3 \wedge v_0 = |escrutinioDip| - d\}\, C_3\, \{|escrutinioDip| - d < v_0\}$

Ahora veamos si $Qc_3 \longrightarrow Post$

Necesitamos probar que $Qc_3 \longrightarrow wp(S_7, Post)$

$S_7 \equiv$ `if` $((votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente))$ `then` $res = false$ `else` $res = true$

$Qc_3 \equiv |escrutinioPres| > 0 \wedge |escrutinioSen| > 0 \wedge |escrutinioDip| > 0 \wedge p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge$
$d = |escrutinioDip| \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j]$
$\wedge\, votosDiputado = \sum_{i=0}^{d-1} escrutinioSen[i]$

$Post \equiv res = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$

$cond \equiv (votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente)$

$wp(S_7, Post) \equiv def(cond) \wedge_L ((cond \wedge_L wp(res : res = false, Post)) \vee (\neg cond \wedge wp(res : res = true, Post)))$

$wp(res : res = \text{false}, Post) \equiv def(res = \text{false}) \wedge_L Post^{res}_{res=\text{false}}$
$\equiv \text{true} \wedge_L (\text{false} = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k])$
$\equiv \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k])$

$wp(res : res = \text{true}, Post) \equiv def(res = \text{true}) \wedge_L Post^{res}_{res=\text{true}}$
$\equiv \text{true} \wedge_L (\text{true} = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k])$
$\equiv \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k])$

$wp(res : res = \text{true}, Post) \equiv wp(res : res = \text{false}, Post) \equiv J$

$wp(S_7, Post) \equiv \text{true} \wedge_L (J \wedge ((cond \vee \neg cond))$
$\equiv J \wedge \text{true}$

$wp(S_7, Post) \equiv \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$
Q$c_3 \longrightarrow wp(S_7, Post)$?
$|escrutinioPres| > 0 \wedge |escrutinioSen| > 0 \wedge |escrutinioDip| > 0 \wedge p = |escrutinioPres| \wedge s = |escrutinioSen| \wedge$
$d = |escrutinioDip| \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k] \wedge votosSenador = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j]$
$\wedge\, votosDiputado = \sum_{i=0}^{d-1} escrutinioSen[i] \longrightarrow \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$
$\equiv true$

Habiendo probado:
$* \, Pre \longrightarrow wp(S_1, S_2, S_3, S_4, S_5, S_6, Pc_1)$

$* \, Pc_1 \longrightarrow wp(C_1, Qc_1)$

$* \, Qc_1 \longrightarrow Pc_2$

$* \, Pc_2 \longrightarrow wp(C_2, Qc_2)$

$* \, Qc_2 \longrightarrow Pc_3$

$* \, Pc_3 \longrightarrow wp(C_3, Qc_3)$

$* \, Qc_3 \longrightarrow Post$

Por teo de la monotonia podemos decir que: $Pre \longrightarrow wp(S_1, S_2, S_3, S_4, S_5, S_6, C_1, C_2, C_3, S_7, Post)$
Demostrando así la correctitud del programa

2.3. Ejercicio 3 - Implementación y WP

obtenerSenadoresEnProvincia $\equiv$

```
    int indice1 := 0;
    int valor1 = escrutinio[0];
    int i = 0;
    while (i < escrutinio.size()-1) do
        if escrutinio[i] > valor1 then
            valor1 := escrutinio[i];
            indice1 := i;
        else
            skip;
        endif
        i := i + 1;
    endwhile
    int indice2 := 0;
    int valor2 := escrutinio[0];
    int d := 0;
    while (d < escrutinio.size()-1) do
        if (valor2 < escrutinio[d]) and (escrutinio[d] < valor1) do
            valor2 := escrutinio[d];
            indice2 := d;
        else
            skip;
        endif
       d := d + 1;
    endwhile
    res := (indice1,indice2);
```

Prueba de Correctitud Ej 3

elección de Invariantes para primer ciclo C_1 :

$I_1 \equiv 0 \leq i \leq |escrutinio| - 1 \wedge_L indice1 = k \iff (0 \leq k \leq i) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < i \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \wedge valor1 = escrutinio[indice1]$
$Pc_1 \equiv indice1 = 0 \wedge valor1 = escrutinio[0] \wedge i = 0 \wedge |escrutinio| > 2$
$Qc_1 \equiv i = |escrutinio| - 1 \wedge indice1 = k \iff (0 \leq k \leq |escrutinio| - 1) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < |escrutinio| - 1 \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \wedge valor1 = escrutinio[indice1]$
$B_1 \equiv i < |escrutinio| - 1$
$Fv_1 \equiv |escrutinio| - 1 - i$

Pruebo que $Pc \longrightarrow I$:

$i = 0 \wedge 0 \leq i < |escrutinio| - 1 \longrightarrow 0 < |escrutinio| - 1 \wedge |escrutinio| > 2 \longrightarrow 0 < 1 < |escrutinio|$TRUE
$i = 0 \wedge indice1 = k \iff (0 \leq k \leq i) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < i \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \longrightarrow$
$indice1 = k \iff (0 \leq k \leq 0) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < 0 \longrightarrow_L escrutinio[k] \geq escrutinio[j])$
$\longrightarrow indice1 = k \iff (k = 0) \wedge_L (\forall j : \mathbb{Z})\ (False \longrightarrow_L escrutinio[k] \geq escrutinio[j])$
$\longrightarrow indice1 = k \iff (k = 0) \wedge_L (\forall j : \mathbb{Z})\ (TRUE) \longrightarrow indice1 = k \iff k = 0 \longrightarrow indice1 = 0$ concuerda con Pc – TRUE
$Indice1 = 0 \wedge valor1 = escrutinio[indice1] \longrightarrow valor1 = escrutinio[0]$ — TRUE

Pruebo que$I_1 \wedge \neg B_1 \longrightarrow Qc_1$:

$\neg B_1 \equiv i \geq |escrutinio| - 1$
$i \geq |escrutinio| - 1 \wedge i \leq |escrutinio| - 1 \longrightarrow i = |escrutinio| - 1$ **TRUE**
Ahora quiero probar que:
$indice1 = k \iff (0 \leq k \leq i) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < i \longrightarrow_L escrutinio[k] \geq escrutinio[j])$
$\longrightarrow$
$indice1 = k \iff (0 \leq k \leq |escrutinio| - 1) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < |escrutinio| - 1 \longrightarrow_L escrutinio[k] \geq escrutinio[j])$
y como recién probamos que $i = |escrutinio| - 1...$
$indice1 = k \iff (0 \leq k \leq |escrutinio| - 1) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < |escrutinio| - 1 \longrightarrow_L escrutinio[k] \geq escrutinio[j])$
es igual a Qc!! :D TRUE
$- -$
Ahora quiero probar:
$I_1 \wedge fv_1 \leq 0 \longrightarrow \neg B_1$
$fv_1 \equiv |escrutinio| - 1 - i \leq 0 \longrightarrow |escrutinio| - 1 \leq i \longrightarrow i \geq |escrutinio| - 1 \equiv \neg B_1$ TRUE
$- -$
$* \{I_1 \wedge B_1\} C_1 \{I_1\}$
Queremos probar que $I_1 \wedge B_1 \longrightarrow wp(C_1, I_1)$

$wp(Q, I_1) \equiv wp($ if $escrutinio[i] > valor1$ then $valor1 := escrutinio[i]$ and $indice1 := i$ else skip$)$
$\equiv 0 \leq i < |escrutinio| - 1 \wedge_L ((escrutinio[i] > valor1 \wedge i = k \iff (\exists k : \mathbb{Z})\ (0 \leq k \leq i)(\forall j : \mathbb{Z})\ (0 \leq j < i \wedge k \neq j \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \vee (escrutinio[i] \leq valor1 \wedge I_1)))$

$wp(C_1, I_1) \equiv wp(i := i + 1; wp(Q, I_1)) \equiv 0 \leq i < |escrutinio| - 2 \wedge_L ((escrutinio[i+1] > valor1 \wedge i + 1 = k \iff (\exists k : \mathbb{Z})\ (0 \leq k \leq i+1)(\forall j : \mathbb{Z})\ (0 \leq j < i + 1 \wedge k \neq j \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \vee (escrutinio[i+1] \leq valor1 \wedge {I_1}^i_{i+1})))$
$I_1 \wedge B_1 \equiv 0 \leq i < |escrutinio| - 1 \wedge_L indice1 = k \iff (0 \leq k \leq i) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < i \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \wedge valor1 = escrutinio[indice1]$

$0 \leq i < |escrutinio| - 1 \longrightarrow 0 \leq i < |escrutinio| - 2 \equiv \text{true}$

$I_1 \longrightarrow ((escrutinio[i+1] > valor1 \wedge i + 1 = k \iff (\exists k : \mathbb{Z})\ (0 \leq k \leq i+1)(\forall j : \mathbb{Z})\ (0 \leq j < i + 1 \wedge k \neq j \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \vee (escrutinio[i+1] \leq valor1 \wedge {I_1}^i_{i+1}))) \equiv \text{true}$

Por lo tanto probamos que: $I_1 \wedge B_1 \longrightarrow wp(C_1, I_1)$

$- - - - - - - - - - - - -$
$* \{I_1 \wedge B_1 \wedge v_0 = |escrutinio| - 1 - i\} C_2 \{|escrutinio| - 1 - i < v_0\}$
Debemos probar que $I_1 \wedge B_1 \wedge v_0 = |escrutinio| - 1 - i \longrightarrow wp(C_1, |escrutinio| - 1 - i < v_0)$

$Q \equiv$ if $escrutinio[i] > valor1$ then $valor1 := escrutinio[i]$ and $indice1 := i$ else skip

$wp(Q, |escrutinio| - 1 - i < v_0) \equiv 0 \leq i < |escrutinio| - 1 \wedge_L$
$((escrutinioi] > valor[i] \wedge |escrutinio| - 1 - i) \vee (escrutinioi] \leq valor[i] \wedge |escrutinio| - 1 - i))$

wp(i:=i+1,Q) $\equiv wp(C_1, |escrutinio| - 1 - i < v_0) \equiv -1 \leq i < |escrutinio| - 2 \wedge_L$
$((escrutinio[i+1] > valor[i+1] \wedge |escrutinio| - i - 2 < v_0) \vee (escrutinio[i+1] \leq valor[i+1] \wedge |escrutinio| - i - 2 < v_0))$

$I_1 \wedge B_1 \wedge v_0 = |escrutinio| - 1 - i \equiv 0 \leq i < |escrutinio| - 1 \wedge_L indice1 = k \iff (0 \leq k \leq i) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < i \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \wedge valor1 = escrutinio[indice1] \wedge v_0 = |escrutinio| - 1 - i$

$0 \leq i < |escrutinio| - 1 \longrightarrow 0 \leq i < |escrutinio| - 2 \equiv \text{true}$

$v_0 = |escrutinio| - 1 - i \longrightarrow |escrutinio| - i - 2 < v_0 \longrightarrow |escrutinio| - i - 2 < |escrutinio| - 1 - i \equiv \text{true}$

Asi probamos que: $\{I_1 \wedge B_1 \wedge v_0 = |escrutinio| - 1 - i\} C_2 \{|escrutinio| - 1 - i < v_0\}$

Planteamos la PreCondición y la Post Condición del Segundo Ciclo:

$Pc_2 \equiv indice2 = 0 \wedge valor2 = escrutinio[0] \wedge d = 0 \wedge |escrutinio| > 2 \wedge i = |escrutinio| - 1 \wedge indice1 = k \iff (0 \leq k \leq$

$|escrutinio| - 1) \wedge_L$

2.4. Ejercicio 6: validarListasEnProvincia

proc validarListasEnProvincia (in cantBancas: $\mathbb{Z}$, in listas: $seq\langle seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle\rangle$) : Bool
requiere $\{cantBancas > 0 \wedge |listas| > 0\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \wedge_L |listas[i]| = CantBancas \wedge cumpleAlternancia(listas[i])) \iff res = \text{true}\}$

pred cumpleAlternancia (in lista: $seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |lista| - 2 \longrightarrow_L lista_1[i] \neq lista_1[i+1])$
}

3. Implementaciones y demostraciones de correctitud

3.1. Ejercicio 1 - Implementación

—dHont— = —escrutinio—-1 $\wedge_L (\forall i : \mathbb{Z})\ (0 \leq ii < |escrutinio| - 1)(|dHont[ik]| = cantBancas) \wedge_L$
$(\forall i, j : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1 \wedge 0 \leq j < cantBancas) dHont[i][j] = \frac{escrutinio[i]}{j+1}$

3.2. Ejercicio 6: validarListasEnProvincia

proc validarListasEnProvincia (in cantBancas: $\mathbb{Z}$, in listas: $seq\langle seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle\rangle$) : Bool
requiere $\{cantBancas > 0 \wedge |listas| > 0\}$
asegura $\{(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \wedge_L |listas[i]| = CantBancas \wedge cumpleAlternancia(listas[i])) \iff res = \text{true}\}$

pred cumpleAlternancia (in lista: $seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |lista| - 2 \longrightarrow_L lista_1[i] \neq lista_1[i+1])$
}

4. Implementaciones y demostraciones de correctitud

4.1. Ejercicio 1 - Implementación

hayBallotage ≡

```
int i := 0;
int suma := 0;
while (i < escrutinio.size()) do
    suma := suma + escrutinio[i];
    i:= i + 1;
endwhile
int j := 0;
bool res := true;
int k := 0;
float porc := 100 / suma;
while (j < escrutinio.size() − 1) do
    if  (escrutinio[j] ∗ porc) > 45 then
        res := false;
    else
        skip;
    endif
    if (escrutinio[j] ∗ porc) <= 45 and (escrutinio[j] ∗ porc) > 40 then
        while (k < escrutinio.size() −1) do
            if (escrutunio[j] ∗ porc) − 10 > (escrutinio[k] ∗ porc) then
                res := false;
            else
                skip;
            endif
            k := k +1;
        endwhile
    else
        skip;
    endif
    j += j + 1;
endwhile
```

4.2. Ejercicio 2 - Implementación y WP

hayFraude ≡

```
int p := 0;
int s := 0;
int d := 0;
int votosPresidente := 0;
int votosSenador := 0;
int votosDiputado := 0;
while p < escrutinioPres.size() do
    votosPresidente := votosPresidente + escrutinioPres[p];
    p := p+1;
endwhile;
while s < escrutinioSen.size() do
    votosSenador := votosSenador + escrurinioSen[s];
    s = s+1;
endwhile;

while d < escrutinioDip.size() do
    votosDiputado := votosDiputado + escrutinioDip[d];
    d := d+1;
endwhile;

if (votosDiputado = votosSenador) and (votosDiputado = votosPresidente) then
    res := false;
else
    res := true;
endif;
```

Prueba de correctitud del programa

Pre $\equiv |escrutinioPres| \geq 1 \wedge |escrutinioDip| \geq 1 \wedge |escrutinioSen| \geq 1$

$S_1 \equiv p = 0$

$S_2 \equiv i = 0$

$S_3 \equiv d = 0$

$S_4 \equiv votosPresidente = 0$

$S_5 \equiv votosSenador = 0$

$S_6 \equiv votosDiputado = 0$

$S_7 \equiv$ `if` $((votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente))$ `then` $res = false$ `else` $res = true$

$Post \equiv res = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$

C_1

$I_1 \equiv 0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0 \wedge votosSen = 0 \wedge votosDip = 0$

$Pc_1 \equiv p = 0 \wedge s = 0 \wedge d = 0 \wedge votosPresidente = 0 \wedge votosDiputado = 0 \wedge votosSenador = 0 \wedge |escrutinioPresDipSen| > 0$

$Qc_1 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$

hayBallotage ≡

```
        int i := 0;
        int suma := 0;
        while (i < escrutinio.size()) do
            suma := suma + escrutinio[i];
            i:= i + 1;
        endwhile
        int j := 0;
        bool res := true;
        int k := 0;
        float porc := 100 / suma;
        while (j < escrutinio.size() − 1) do
            if  (escrutinio[j] ∗ porc) > 45 then
                res := false;
            else
                skip;
            endif
            if (escrutinio[j] ∗ porc) <= 45 and (escrutinio[j] ∗ porc) > 40 then
                while (k < escrutinio.size() −1) do
                    if (escrutunio[j] ∗ porc) − 10 > (escrutinio[k] ∗ porc) then
                        res := false;
                    else
                        skip;
                    endif
                    k := k +1;
                endwhile
            else
                skip;
            endif
            j += j + 1;
        endwhile
```

4.3. Ejercicio 2 - Implementación y WP

hayFraude ≡

```
    int p := 0;
    int s := 0;
    int d := 0;
    int votosPresidente := 0;
    int votosSenador := 0;
    int votosDiputado := 0;
    while p < escrutinioPres.size() do
        votosPresidente := votosPresidente + escrutinioPres[p];
        p := p+1;
    endwhile;
    while s < escrutinioSen.size() do
        votosSenador := votosSenador + escrurinioSen[s];
        s = s+1;
    endwhile;

    while d < escrutinioDip.size() do
        votosDiputado := votosDiputado + escrutinioDip[d];
        d := d+1;
    endwhile;

    if (votosDiputado = votosSenador) and (votosDiputado = votosPresidente) then
        res := false;
    else
        res := true;
    endif;
```

Prueba de correctitud del programa

Pre $\equiv |escrutinioPres| \geq 1 \wedge |escrutinioDip| \geq 1 \wedge |escrutinioSen| \geq 1$
$S_1 \equiv p = 0$
$S_2 \equiv i = 0$
$S_3 \equiv d = 0$
$S_4 \equiv votosPresidente = 0$
$S_5 \equiv votosSenador = 0$
$S_6 \equiv votosDiputado = 0$
$S_7 \equiv$ `if` $((votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente))$ `then` $res = false$ `else` $res = true$
$Post \equiv res = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$
$|dHont| = |escrutinio| - 1 \wedge_L (\forall i : \mathbb{Z})\ (0 \leq ii < |escrutinio| - 1)(|dHont[ik]| = cantBancas) \wedge_L$
$(\forall i, j : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1 \wedge 0 \leq j < cantBancas) dHont[i][j] = \frac{escrutinio[i]}{j+1}$

4.4. Ejercicio 6: validarListasEnProvincia

`proc validarListasEnProvincia` (in cantBancas: $\mathbb{Z}$, in listas: $seq\langle seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle\rangle$) : Bool
    `requiere` $\{cantBancas > 0 \wedge |listas| > 0\}$
    `asegura` $\{(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \wedge_L |listas[i]| = CantBancas \wedge cumpleAlternancia(listas[i])) \iff res = \text{true}\}$

`pred cumpleAlternancia` (in lista: $seq\langle dni : \mathbb{Z} \times genero : \mathbb{Z}\rangle$) {
    $(\forall i : \mathbb{Z})\ (0 \leq i < |lista| - 2 \longrightarrow_L lista_1[i] \neq lista_1[i+1])$
}

5. Implementaciones y demostraciones de correctitud

5.1. Ejercicio 1 - Implementación

hayBallotage ≡

```
int i := 0;
int suma := 0;
while (i < escrutinio.size()) do
    suma := suma + escrutinio[i];
    i:= i + 1;
endwhile
int j := 0;
bool res := true;
int k := 0;
float porc := 100 / suma;
while (j < escrutinio.size() − 1) do
    if (escrutinio[j] ∗ porc) > 45 then
        res := false;
    else
        skip;
    endif
    if (escrutinio[j] ∗ porc)<= 45 and (escrutinio[j] ∗ porc) > 40 then
        while (k < escrutinio.size() −1) do
            if (escrutunio[j] ∗ porc) − 10 > (escrutinio[k] ∗ porc) then
                res := false;
            else
                skip;
            endif
            k := k +1;
        endwhile
    else
        skip;
    endif
    j += j + 1;
endwhile
```

5.2. Ejercicio 2 - Implementación y WP

hayFraude ≡

```
int p := 0;
int s := 0;
int d := 0;
int votosPresidente := 0;
int votosSenador := 0;
int votosDiputado := 0;
while p < escrutinioPres.size() do
    votosPresidente := votosPresidente + escrutinioPres[p];
    p := p+1;
endwhile;
while s < escrutinioSen.size() do
    votosSenador := votosSenador + escrurinioSen[s];
    s = s+1;
endwhile;

while d < escrutinioDip.size() do
    votosDiputado := votosDiputado + escrutinioDip[d];
    d := d+1;
endwhile;

if (votosDiputado = votosSenador) and (votosDiputado = votosPresidente) then
    res := false;
else
    res := true;
endif;
```

Prueba de correctitud del programa

Pre $\equiv |escrutinioPres| \geq 1 \wedge |escrutinioDip| \geq 1 \wedge |escrutinioSen| \geq 1$

$S_1 \equiv p = 0$

$S_2 \equiv i = 0$

$S_3 \equiv d = 0$

$S_4 \equiv votosPresidente = 0$

$S_5 \equiv votosSenador = 0$

$S_6 \equiv votosDiputado = 0$

$S_7 \equiv$ `if` $((votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente))$ `then` $res = false$ `else` $res = true$

$Post \equiv res = \text{false} \iff$

$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$

C_1

$I_1 \equiv 0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0 \wedge votosSen = 0 \wedge votosDip = 0$

$Pc_1 \equiv p = 0 \wedge s = 0 \wedge d = 0 \wedge votosPresidente = 0 \wedge votosDiputado = 0 \wedge votos|dHont| = |escrutinio| - 1 \wedge_L (\forall i : \mathbb{Z})\ (0 \leq ii < |escrutinio| - 1)(|dHont[ik]| = cantBancas) \wedge_L$

$(\forall i, j : \mathbb{Z})\ (0 \leq i < |escrutinio| - 1 \wedge 0 \leq j < cantBancas) dHont[i][j] = \frac{escrutinio[i]}{j+1}$

5.3. Ejercicio 6: validarListasEnProvincia

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

`requiere` $\{cantBancas > 0 \wedge |listas| > 0\}$

`asegura` $\{(\forall i : \mathbb{Z})\ (0 \leq i < |listas| \wedge_L |listas[i]| = CantBancas \wedge cumpleAlternancia(listas[i])) \iff res = \text{true}\}$

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

$(\forall i : \mathbb{Z})\ (0 \leq i < |lista| - 2 \longrightarrow_L lista_1[i] \neq lista_1[i+1])$

}

6. Implementaciones y demostraciones de correctitud

6.1. Ejercicio 1 - Implementación

hayBallotage ≡

```
int i := 0;
int suma := 0;
while (i < escrutinio.size()) do
    suma := suma + escrutinio[i];
    i:= i + 1;
endwhile
int j := 0;
bool res := true;
int k := 0;
float porc := 100 / suma;
while (j < escrutinio.size() − 1) do
    if  (escrutinio[j] ∗ porc) > 45 then
        res := false;
    else
        skip;
    endif
    if (escrutinio[j] ∗ porc) <= 45 and (escrutinio[j] ∗ porc) > 40 then
        while (k < escrutinio.size() −1) do
            if (escrutunio[j] ∗ porc) − 10 > (escrutinio[k] ∗ porc) then
                res := false;
            else
                skip;
            endif
            k := k +1;
        endwhile
    else
        skip;
    endif
    j += j + 1;
endwhile
```

6.2. Ejercicio 2 - Implementación y WP

hayFraude ≡

```
int p := 0;
int s := 0;
int d := 0;
int votosPresidente := 0;
int votosSenador := 0;
int votosDiputado := 0;
while p < escrutinioPres.size() do
    votosPresidente := votosPresidente + escrutinioPres[p];
    p := p+1;
endwhile;
while s < escrutinioSen.size() do
    votosSenador := votosSenador + escrurinioSen[s];
    s = s+1;
endwhile;

while d < escrutinioDip.size() do
    votosDiputado := votosDiputado + escrutinioDip[d];
    d := d+1;
endwhile;

if (votosDiputado = votosSenador) and (votosDiputado = votosPresidente) then
    res := false;
else
    res := true;
endif;
```

Prueba de correctitud del programa

Pre $\equiv |escrutinioPres| \geq 1 \wedge |escrutinioDip| \geq 1 \wedge |escrutinioSen| \geq 1$

$S_1 \equiv p = 0$

$S_2 \equiv i = 0$

$S_3 \equiv d = 0$

$S_4 \equiv votosPresidente = 0$

$S_5 \equiv votosSenador = 0$

$S_6 \equiv votosDiputado = 0$

$S_7 \equiv \texttt{if}\ ((votosDiputado = votosSenador) \wedge (votosDiputado = votosPresidente))\ \texttt{then}\ res = false\ \texttt{else}\ res = true$

$Post \equiv res = \text{false} \iff$
$\sum_{i=0}^{|escrutinioPres|-1} escrutinioPres[i] = \sum_{j=0}^{|escrutinioSen|-1} escrutinioSen[j] = \sum_{k=0}^{|escrutinioDip|-1} escrutinioDip[k]$

C_1

$I_1 \equiv 0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0 \wedge votosSen = 0 \wedge votosDip = 0$

$Pc_1 \equiv p = 0 \wedge s = 0 \wedge d = 0 \wedge votosPresidente = 0 \wedge votosDiputado = 0 \wedge votosSenador = 0 \wedge |escrutinioPresDipSen| > 0$

$Qc_1 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$

$Senador = 0 \wedge |escrutinioPresDipSen| > 0$

$Qc_1 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$

C$_1$

$I_1 \equiv 0 \leq p \leq |escrutinioPres| \wedge votosPresidente = \sum_{k=0}^{p-1} escrutinioPres[k] \wedge s = 0 \wedge d = 0 \wedge votosSen = 0 \wedge votosDip = 0$

$Pc_1 \equiv p = 0 \wedge s = 0 \wedge d = 0 \wedge votosPresidente = 0 \wedge votosDiputado = 0 \wedge votosSenador = 0 \wedge |escrutinioPresDipSen| > 0$

$Qc_1 \equiv p = |escrutinioPres| \wedge s = 0 \wedge d = 0 \wedge votosPresidente = \sum_{k=0}^{|escrutioPres|-1} escrutinioPres[k]$

$\mathbb{Z} 0 \leq j < |escrutinio| - 1 \longrightarrow_L escrutinio[k] \geq escrutinio[j] \wedge valor1 = escrutinio[indice1]$

$Qc_2 \equiv d = |escrutinio| - 1 \wedge i = |escrutinio| - 1 \wedge indice2 = h \iff (0 \leq h \leq |escrutinio| - 1) \wedge_L (\forall s : \mathbb{Z})\ (0 \leq s < |escrutinio| - 1 \longrightarrow_L escrutinio[h] \geq escrutinio[s]) \wedge indice1 = k \iff (0 \leq k \leq |escrutinio| - 1) \wedge_L (\forall j : \mathbb{Z})\ (0 \leq j < |escrutinio| - 1 \longrightarrow_L escrutinio[k] \geq escrutinio[j]) \wedge valor2 = escrutinio[indice2] valor1 = escrutinio[indice1]$

\- -

Podemos ver que finalmente llegaremos a que:

$Qc_1 \longrightarrow Pc_2$

$\wedge$
$Pc_2 \longrightarrow Qc_2$
$\wedge$
$Qc_2 \longrightarrow Post$

Post : $res_0 = indice1 \wedge \ res_1 = indice2$

Por Teo de la Monotonia demostramos la correctitud del programa, y hallamos: $wp(S_1, C_1, S_2, C_2, S_3, Post)$

6.3. Ejercicio 4 - Implementación

404

6.4. Ejercicio 5 - Implementación

404

6.5. Ejercicio 6 - Implementación

validarListasEnProvincia $\equiv$

```
bool res := true;
int i := 0;
int j := 0;
while i < listas.size() do
    if cantBancas != listas[i].size() then
        res:= false;
    else
        skip;
    while j < listas[i].size() − 1 do
        if lista[i][j] = lista[i][j+1] then
            res := false;
        else
            skip;
        j = j + 1;
    endwhile
    i = i + 1;
endwhile
```