

Nro. de orden: 9
 LU: 526/15
 Apellidos: FREUND
 Nombres: TEODORO

1		2	3	4	5		TOTAL
a	b				a	b	
15		20	20	24	15		94

(A)

Aclaraciones: El parcial NO es a libro abierto. Cualquier decisión de interpretación que se tome debe ser aclarada y justificada. Para aprobar se requieren al menos 60 puntos. **Entregar cada ejercicio en hoja separada.**

Importante: Para la resolución del parcial NO es necesario ni está permitido el uso de **acum**. En caso de ser necesario, pueden asumir como definida la función **sum** (Σ) sobre listas. Las funciones **min** o **max** sobre listas, en caso de requerirlas, deben ser definidas.

```

tipo Nombre=String;
tipo Bloque=String;
tipo Provincia = Buenos Aires, Catamarca, Chaco,...;
tipo Comision = Cultura, Educacion, Finanzas,...;
tipo TipoMayoría = Simple, Absoluta;

tipo Diputado {
    observador nombre (d: Diputado) : Nombre;
    observador bloque (d: Diputado) : Bloque;
    observador distrito (d: Diputado) : Provincia;
    observador comisiones (d: Diputado) : [Comision];
    invariante comisionesOrdenadas : ordenada(comisiones(d));
}

tipo Proyecto {
    observador numero (p: Proyecto) : Z;
    observador descripcion (p: Proyecto) : String;
    observador autores (p: Proyecto) : [Diputado];
    observador tratadoEn (p: Proyecto) : Comision;
    observador aprobacion (p: Proyecto) : TipoMayoría;
    invariante autoresOrdenados : ordenada(nombres(autores(p)));
    invariante autoresEnComision : ...;
}

tipo Sesion {
    observador diputados (s: Sesion) : [Diputado];
    observador ordenDelDia (s: Sesion) : [Proyecto];
    observador presente (s: Sesion, n: Nombre) : Bool;
        requiere n ∈ nombres(diputados(s));
    observador voto (s: Sesion, nroP: Z, n: Nombre) : Bool;
        requiere nroP ∈ numeros(ordenDelDia(s)) ∧
            n ∈ nombres(diputados(s)) ∧ presente(s, n);
    invariante diputadosDistintos :
        sinRepetidos(nombres(diputados(s)));
    invariante proyectosDistintos :
        sinRepetidos(numeros(ordenDelDia(s)));
    invariante camaraBienConstituida : ...;
}

aux numeros (ps: [Proyecto]) : [Z] = [numero(p) | p ← ps];
aux nombres (ds: [Diputado]) : [Nombre] = [nombre(d) | d ← ds];
aux sinRepetidos (xs: [T]) : Bool = (∀i, j ← [0..|xs|], i ≠ j) xs_i ≠ xs_j;
aux ordenada (xs: [T]) : Bool = (∀i ← [0..|xs| - 1]) xs_i ≤ xs_{i+1};
    
```

Ejercicio 1. [15 puntos]

- Completar el invariante `autoresEnComision` del tipo `Proyecto`, que indica que el proyecto tiene al menos un autor, y que todos los diputados que escribieron el proyecto forman parte de la comisión en la cual se trata dicho proyecto.
- Completar el invariante `camaraBienConstituida` del tipo `Sesion`, que expresa que hay al menos un diputado de cada uno de los 24 distritos.

Ejercicio 2. [20 puntos] Especificar el problema `elMasTraidor (s: Sesion) = result : Nombre` que devuelve el nombre de alguno de los diputados que más proyectos traicionó. Diremos que un diputado *traiciona* un proyecto cuando es uno de los autores del mismo pero lo vota de forma negativa.

Ejercicio 3. [20 puntos] Especificar el problema `losRechazados (ss: [Sesion]) = result : [Z]` que devuelve una lista con los números de los proyectos no aprobados en alguna sesión de `ss`. La lista resultante debe estar ordenada de manera creciente y no contener repetidos. Considerar que en distintas sesiones no pueden tratarse proyectos diferentes que tengan el mismo número. La aprobación de un proyecto depende del tipo de mayoría:

- Si requiere mayoría simple, debe tener al menos la mitad de los votos de los diputados presentes.
- Si requiere mayoría absoluta, la cantidad de votos debe ser mayor a la mitad de la cantidad total de diputados.

Aclaración: Si un proyecto es tratado en más de una sesión de `ss`, y en algunas se aprueba y en otras no, el proyecto debe estar en la lista devuelta como resultado del problema.

Ejercicio 4. [30 puntos] Especificar el problema `renunciarAComision (s: Sesion, d: Diputado, c: Comision)`, que modifica la sesión de forma tal que el diputado `d` ya no forme parte de la comisión `c`.

Ejercicio 5. [15 puntos] Dado el siguiente programa:

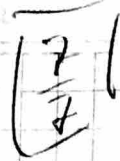
```

int restarONo(int a, int b){
    int result = 0;
    if(a > b){
        result = a - b;
    }
    return result;
}
    
```

- Dar la transformación de estados del programa presentado.
- Decidir si el programa es correcto para las siguientes especificaciones. En caso afirmativo, demostrar. En caso contrario, justificar por qué no es correcto.

problema restarONo (a,b: Z) = res : Z { requiere a == b; asegura res == 0; }	problema restarONo (a,b: Z) = res : Z { asegura a > b ⇒ res == a - b; asegura a ≤ b ⇒ res == 0; }	problema restarONo (a,b: Z) = res : Z { asegura res ≥ 0; }
---	--	--

① a.



invariante autores En Comision: $|autores(P)| > 0 \wedge$
 $(\forall d \leftarrow autores(P)) \text{ tratado En}(P) \in comisiones(d);$

b.

invariante cámara Bien Constituida:

$(\forall i \leftarrow [0..24]) (\exists d \leftarrow diputados(s)) \text{ ord}(\text{distrito}(d)) = i;$



ORDEN 9

②

problema el Mas Traidor (s: Sesion) = result: Nombre {

~~asegura result ∈ nombres (diputados (s))~~

asegura result ∈ nombres (diputados (s));

asegura (∀ d ∈ diputados (s)) Traiciones (s, d) ∈

Traiciones (s, DipNombre (s, result));

∃ x Traiciones (s: Sesion, d: Diputado): $\mathbb{Z} =$

[[20] p ← orden Del Dia (s), d ∈ autores (p),
presente (s, nombre (d)), ¬ voto (s, numero (p), nombre (d))]];

∃ x DipNombre (s: Sesion, n: Nombre): Diputado =

cab ([d] d ∈ diputados (s), n = nombre (d))];

}

(SE UTILIZA DipNombre YA QUE EN EL TIPO PROYECTO
PUEDE HABER ALGUN DIPUTADO QUE SE LLAME IGUAL QUE
OTRO DIP. DISTINTA, QUE TORNE PARTE DE LA SESION

3

Problema los Rechazados (ss: [Sesion]) = result: [Z] {

requiere números Únicos: ($\forall s, s' \leftarrow ss, p \leftarrow \text{ordenDelDia}(s),$

$p' \leftarrow \text{ordenDelDia}(s') \text{ numero}(p) = \text{numero}(p') \Rightarrow p = p'$;

~~Figura de los Rechazados (P, P')~~ ✓

asegura ~~mis~~ mismos(result, Proy Rech(ss)); ✓

asegura ordenada(result); ✓

aux Borrar Rep(xs: [T]): [T] = [xs[i] | i ∈ [0..|xs|], xs[i] ≠ xs[0..i-1]]

aux Proy Rech(ss: [Sesion]): [Z] =

BorrarRep([numero(p) | s ∈ ss, p ∈ ordenDelDia(s),
Rechazado(s, p)]);

aux Rechazado(s: Sesion, p: Proyecto): Bool =

(aprobacion(p) = Simple ∧ DipPresentes(s) > ~~2 *~~

2 * Votos(p, s, DipPresentes(s)))

∨ (aprobacion(p) = Absoluta ∧ Diputados(s) ≥

2 * Votos(p, s, DipPresentes(s)))

aux DipPresentes(s: Sesion): [Diputado] =

[d | d ∈ Diputados(s), presente(s, nombre(d))]; ✓

aux Votos(p: Proyecto, s: Sesion, ds: [Diputado]): Z =

[Z0 | d ∈ ds, voto(s, numero(p), nombre(d))]; ✓

4

Problema renunciar A Comision (s: Sesion, d: Diputado, c: Comision) {
 requiere $(\forall p \in \text{orden Del Dia}(s), \text{tratado En}(p) = c, \text{de autores}(p))$
 $|\text{autores}(p)| > 1$;

modifica s;

asegura $|\text{diputados}(s)| = |\text{diputados}(\text{pre}(s))|$;

asegura $|\text{orden Del Dia}(s)| = |\text{orden Del Dia}(\text{pre}(s))|$;

asegura $(\forall d' \in \text{diputados}(\text{pre}(s)), d' \neq d)$

$(\exists d'' \in \text{diputados}(s)) d' = d''$;

asegura $(\forall d' \in \text{diputados}(\text{pre}(s)), d' = d) (\exists d'' \in \text{diputados}(s))$

$\text{nombre}(d') = \text{nombre}(d'')$ \wedge

$\text{bloque}(d') = \text{bloque}(d'')$ \wedge

$\text{distrito}(d') = \text{distrito}(d'')$ \wedge

$\text{mismas}(\text{comisiones}(d'), c: \text{comisiones}(d''))$;

asegura $(\forall p \in \text{orden Del Dia}(\text{pre}(s)), \text{tratado En}(p) \neq c)$
 $\text{de autores}(p)$

$(\exists p' \in \text{orden Del Dia}(s)) p = p'$;

(SIGUE ATRAS)

¿ Si el proyecto se trató en
 otra comisión pero p era autor?

asegura $(\forall p \leftarrow \text{ordenDelDia}(\text{pre}(s)), \text{tratadoEn}(p) = c \text{ (1,2) } \wedge$
 $\downarrow \in \text{autores}(p))$

$(\exists p' \leftarrow \text{ordenDelDia}(s))$

$\text{numero}(p) = \text{numero}(p') \wedge$

$\text{descripcion}(p) = \text{descripcion}(p') \wedge$

$\text{tratadoEn}(p) = c \wedge$

$\text{aprobacion}(p) = \text{aprobacion}(p') \wedge$

$\text{mismos}(\text{autores}(p), \text{autores}(p'));$ ✓

asegura $(\forall n \leftarrow \text{nombres}(\text{diputados}(s)))$

$\text{presente}(s, n) = \text{presente}(\text{pre}(s), n);$ ✓

~~$(\forall n \leftarrow \text{nombres}(\text{diputados}(s)),$~~

asegura $(\forall n \leftarrow \text{nombres}(\text{diputados}(s));$ ~~$\text{presente}(s, n)$~~

$n \leftarrow \text{numero}(\text{ordenDelDia}(s)), \text{presente}(s, n))$

$\text{voto}(s, n, n) = \text{voto}(\text{pre}(s), n, n);$ ✓

}

5) 3.

```

int restarONo (int a, int b) {
    int result = 0;
    // E1;
    // vale result == 0; (a y b no cambian) ✓
    // Pif: result == 0;
    // implica Pif;
    if (a > b) {
        // E1f0;
        // vale Pif ∧ B; ✓
        // implica result == 0 ∧ a > b;
        result = a - b;
        // E1f1;
        // vale result == a - b ∧ a > b;
        // implica (result == a - b ∧ a > b) ∨ (result == 0 ∧ a ≤ b);
        // implica Qif; → sintacticamente iguales ✓
        // no false?
    }
    // E2;
    // vale Qif: (result == a - b ∧ a > b) ∨ (result == 0 ∧ a ≤ b);
    return result;
    // E3;
    // vale res == result @ E2 ∧ Qif == Qif @ E2;
    // implica (res == a - b ∧ a > b) ∨ (res == 0 ∧ a ≤ b);
}
    
```

(SIGUE ATRÁS)

RAMA FALSE

// $E \in 0$;

// vale $P \text{ if } \neg B$;

// implica $\text{result} == 0 \wedge a \leq b$;

// implica $(\text{result} == a - b \wedge a > b) \vee (\text{result} == 0 \wedge a \leq b)$;

// implica $Q \text{ if } ; \rightsquigarrow$ SINTACTICAMENTE IGUALES

$\rightarrow q \Rightarrow P \vee q$

SI P NO SE
INDEFINIE ✓

~~FALSE~~
(ESTE CASO) (0)

→ NOTAR, PARA TODAS LAS JUSTIFICACIONES, QUE ESTE PROGRAMA SIEMPRE TERMINA

6. problema $\text{restar} \text{ ONO } (a, b: \mathbb{Z}) = \text{res}: \mathbb{Z} \{$

requiere $a == b$;

asegura $\text{res} == 0$;

}

ES CORRECTO YA QUE SI $a == b$, COMO AL FINAL DEL

PROGRAMA VALE QUE $(\text{res} == a - b \wedge a > b) \vee (\text{res} == 0 \wedge a \leq b)$

Y COMO $a > b$ NO VALE, DEBE SER $\text{res} == 0$, CUMPLIENDO

EL ASERCIÓN ✓

problema $\text{restar} \text{ ONO } (a, b: \mathbb{Z}) = \text{res}: \mathbb{Z} \{$

asegura $a > b \Rightarrow \text{res} == a - b$;

asegura $a \leq b \Rightarrow \text{res} == 0$;

}

ES CORRECTO, PORQUE:

SEAN $P = a > b$, $q = \text{res} == a - b$, $q' = \text{res} == 0$

ENTONCES, POR DEFINICIONES DE LAS OPERACIONES LÓGICAS

$$(P \Rightarrow Q) \wedge (\neg P \Rightarrow Q') \Leftrightarrow (\neg P \vee Q) \wedge (P \vee Q')$$

$$(\neg P \wedge \neg P) \vee (\neg P \wedge Q') \vee (Q \wedge P) \vee (Q \wedge Q')$$

$$(\neg P \wedge Q') \vee (Q \wedge P) \Leftrightarrow$$

$$(a \leq b \wedge res == 0) \vee (res == a - b \wedge a > b) \Leftrightarrow$$

$$(res == a - b \wedge a > b) \vee (res == 0 \wedge a \leq b)$$

↳ ES DECIR QUE EL PROBLEMA ASEGURA LO MISMO QUE EL PROGRAMA CUMPLE AL FINALIZAR ✓

Problema restar $QND(a, b: \mathbb{Z}) = res: \mathbb{Z}$ {
asegura $res \geq 0$;
}

NOTAR QUE

$$res == 0 \wedge a \leq b \Rightarrow res \geq 0 \quad \vee$$

$$res == a - b \wedge a > b \Rightarrow res \geq 0$$

~~NOTAR~~ ENTONCES

$$(res == 0 \wedge a \leq b) \vee (res == a - b \wedge a > b) \Rightarrow res \geq 0 \quad \checkmark$$

↳ POR LA REGLA DE LA ELIMINACION DEL \vee ✓

ENTONCES, EL PROGRAMA SIEMPRE DEVUELVE un $res \geq 0$, LO QUE CUMPLE EL asegura DEL PROBLEMA ✓