UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación– Algoritmos y Estructura de Datos I
Recueperatorio del parcial de especificación– 27/11/2013

LU:
Apellidos:
Nombres:
Orden:
Turno:

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**.

```
tipo Cargo = Presidente, Senador, Diputado, Gobernador, Intendente;
tipo Partido=String;
tipo DNI=Z;
tipo Nombre=String;
tipo Politico=(Nombre, Partido);

tipo Boleta {
  observador lista (b: Boleta) : Z;
  observador candidatos (b: Boleta) : [(Politico, Cargo)];
  invariante todosDelMismoPartido :
    (∀c, d ← candidatos(c))sgd(prm(c)) == sgd(prm(d));
  invariante listaValida : lista(b) > 0;
  invariante alMenosUnCandidato : |candidatos(b)| > 0;
  invariante sinRepetidos(primeros(candidatos))
    ∧ sinRepetidos(segundos(candidatos));
}

tipo Mesa {
  observador votantes (m: Mesa) : [DNI];
  observador presidente (m: Mesa) : DNI;
  observador yaVoto (m: Mesa, d: DNI) : Bool;
    requiere d ∈ votantes(m);
  observador voto (m: Mesa, d: DNI) : Boleta;
    requiere d ∈ votantes(m) ∧ yaVoto(m, d);
  invariante DNIsValidos : (∀v ← votantes(m))v > 0;
  invariante sinVotantesRepetidos : sinRepetidos(votantes(m))
  invariante presidenteEsVotanteYVoto : ...;
}

tipo Eleccion {
  observador padron (e: Eleccion) : [DNI];
  observador boletas (e: Eleccion) : [Boleta];
  observador mesas (e: Eleccion) : [Mesa];
  invariante unaBoletaDeCadaPartido :
    sinRepetidos(partidos(boletas(e)));
  invariante politicosDeUnSoloPartido : ...;
  invariante nadieVotaDosVeces : sinRepetidos(padron(e));
  invariante listasDistintas :
    sinRepetidos([lista(b) | b ← boletas(e)]);
  invariante mesasUnicas : (∀i, j ← [0..|mesas(e)|), i ≠ j)
    ¬mesasIguales(mesas(e)_i, mesas(e)_j);
  invariante padronYMesasConsistentes :
    mismos(padron(e), concat[votantes(m) | m ← mesas(e)]);
  invariante votanBoletasDisponibles : (∀m ← mesas(e))
    (∀v ← votantes(m))yaVoto(m, v) ⇒ boletaValida(voto(m, v),
}

aux sinRepetidos (l: [T]) : Bool = (∀i ← [0..|l|)) l_i ∉ l(i..|l|);
aux primeros (l: [(T,U)]) : [T] = [prm(x) | x ← l];
aux segundos (l: [(T,U)]) : [U] = [sgd(x) | x ← l];
aux partidos (bs: [Boleta]) : [Partido] = [sgd(prm(candidatos(b)_0)) | b ← bs];
aux mesasIguales (m1, m2: Mesa) : Bool = mismos(votantes(m1), votantes(m2)) ∧
  votaronLosMismos(m1, m2) ∧ votaronLoMismo(m1, m2) ∧ presidente(m1) == presidente(m2)
aux votaronLosMismos (m1, m2: Mesa) : Bool = (∀v ← votantes(m1))yaVoto(m1, v) ⇔ yaVoto(m2, v);
aux votaronLoMismo (m1, m2: Mesa) : Bool = (∀v ← votantes(m1), yaVoto(m1, v))boletasIguales(voto(m1, v), voto(m2, v));
aux boletaValida (b: Boleta, e: Eleccion) : Bool = (∃x ← boletas(e))boletasIguales(b, x);
aux boletasIguales (b1, b2: Boleta) : Bool = lista(b) == lista(x) ∧ mismos(candidatos(b), candidatos(x));
```

Ejercicio 1. [30 puntos]

a) [10 p.] Especificar el invariante `presidenteEsVotanteYVoto`, que indica que el presidente de la mesa es votante de la misma y emitió su voto.
b) [10 p.] Especificar el invariante `politicosDeUnSoloPartido`, que indica que no hay políticos que formen parte de boletas de distintos partidos.
c) [10 p.] Especificar el `aux cantidadVotos (m: Mesa, n: Nombre) = Z;`, que devuelve la cantidad de votos que tuvo el candidato de nombre n entre las personas que ya votaron en la mesa m.

Ejercicio 2. [25 puntos]

Especificar el `problema ganadores (e: Eleccion, c: Cargo) = result : [Politico]` , que devuelve todos aquellos políticos que se postularon para el cargo c que mayor cantidad de votos hayan recibido. La lista devuelta debe estar ordenada de acuerdo al nombre de los políticos.

Ejercicio 3. [30 puntos]

Especificar el `problema reemplazarPresidentes (e: Eleccion, m: [(Mesa, DNI)])` . Este problema modifica la elección reemplazando a los presidentes de las mesas incluídas en la lista pasada como parámetro por las personas indicadas en dicha lista.

Ejercicio 4. [15 puntos]

(Ejercicio 23.1 de la de la práctica 4)

Especificar el `problema escaleraMasLarga (l:[Z]) = result : (Z, Z)` , que devuelve las posiciones de inicio y fin de la subcadena de l que constituya la escalera más larga. Una escalera es una secuencia de números donde cada uno es el sucesor del anterior. Por ejemplo: