UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructura de Datos I

Primer cuatrimestre de 2012

30 de Marzo de 2012

TPE - OJOTA (Organización de Juegos Olímpicos Tp de Algoritmos 1) v1.0

Aclaraciones

- Para aprobar la totalidad del TP es necesario tener aprobado cada uno de sus módulos.
- **No** está permitido el uso de **acum** para la resolución.

El Comité Olimpico Internacional (COI) decide instalar su oficina en nuestra país, fundando su decisión en que Argentina es un país con Buena Gente. Luego de barajar distintas opciones resuelve contratar nuestros servicios para que modelemos ciertos problemas relacionados a los Juegos Olímpicos. Después de una serie de reuniones hemos podido recabar suficiente información acerca del problema a resolver, tomando las decisiones de diseño que se detallan a continuación. Nuestra misión, si decidimos aceptarla, es especificar todo lo que se pide a continuación.

1. Tipos

`tipo` Deporte = String;
`tipo` Pais = String;
`tipo` Sexo = Femenino, Masculino;

2. Atleta

`tipo Atleta` {
  `observador nombre` (a: Atleta) : String;
  `observador sexo` (a: Atleta) : Sexo;
  `observador añoNacimiento` (a: Atleta) : $\mathbb{Z}$;
  `observador nacionalidad` (a: Atleta) : Pais;
  `observador ciaNumber` (a: Atleta) : $\mathbb{Z}$;
  `observador deportes` (a: Atleta) : [Deporte];
  `observador capacidad` (a: Atleta, d: Deporte) : $\mathbb{Z}$;
      `requiere` $d \in deportes(a)$;
  `invariante` $sinRepetidos(deportes(a))$;
  `invariante` $ordenada(deportes(a))$;
  `invariante capacidadEnRango` : $(\forall d \leftarrow deportes(a))0 \leq capacidad(a, d) \leq 100$;
}

1. `problema entrenarNuevoDeporte` (a: Atleta, d: Deporte, c: $\mathbb{Z}$)
   Modifica al atleta a agregando al deporte d entre los deportes que practica. c representa la capacidad del atleta en ese deporte.

3. Competencia

`tipo Competencia` {
  `observador categoria` (c: Competencia) : (Deporte, Sexo);
  `observador participantes` (c: Competencia) : [Atleta];
  `observador finalizada` (c: Competencia) : Bool;
  `observador ranking` (c: Competencia) : [Atleta];
      `requiere` $finalizada(c)$;
  `observador lesTocoControlAntiDoping` (c: Competencia) : [Atleta];
      `requiere` $finalizada(c)$;
  `observador leDioPositivo` (c: Competencia, a: Atleta) : Bool;
      `requiere` $finalizada(c) \wedge a \in lesTocoControlAntiDoping(c)$;
  `invariante participaUnaSolaVez` : $sinRepetidos(ciaNumbers(participantes(c)))$;
  `invariante participantesPertenecenACat` :
    $(\forall p \leftarrow participantes(c))prm(categoria(c)) \in deportes(p) \wedge sgd(categoria(c)) == sexo(p)$;

```
  invariante elRankingEsDeParticipantesYNoHayRepetidos :
      finalizada(c) ⇒ incluida(ranking(c), participantes(c)) ;
  invariante seControlanParticipantesYNoHayRepetidos :
      finalizada(c) ⇒ incluida(lesTocoControlAntiDoping(c), participantes(c)) ;
}
```

2. `problema finalizarCompetencia` (c: Competencia, posiciones: [Atleta], control: [(Atleta, Bool)])
   Registra que una competencia terminó. La lista *posiciones* contiene a todos los participantes que finalizaron la competencia, en el orden en el que finalizaron. La lista *control* indica qué participantes debieron someterse al control antidopping, y cuál fue el resultado obtenido.
3. `problema linfordChristie` (c: Competencia, a: Atleta)
   Modifica la competencia c, descalificando al atleta a de la competencia. Como este problema es para inconvenientes durante la competencia, la misma no debe estar finalizada.
4. `problema gananLosMasCapaces` (c: Competencia) = `result` : Bool
   Indica si entre los participanets que finalizaron la competencia c, los que obtuvieron los mejores resultados eran los que más capacidad tenían para el deporte de dicha competencia.
5. `problema sancionarTramposos` (c: Competencia)
   Modifica una competencia eliminando del ranking a aquellos participantes a quienes el control antidoping les dio positivo.

4. JJOO

```
tipo JJOO {
  observador año (j: JJOO) : ℤ ;
  observador atletas (j: JJOO) : [Atleta] ;
  observador cantDias (j: JJOO) : ℤ ;
  observador cronograma (j: JJOO, dia: ℤ) : [Competencia] ;
       requiere 1 ≤ dia ≤ cantDias(j) ;
  observador jornadaActual (j: JJOO) : ℤ ;

  invariante atletasUnicos : sinRepetidos(ciaNumbers(atletas(j))) ;
  invariante unaDeCadaCategoria : (∀i, k ← [0.. |competencias(j)|), i ≠ k)
     categoria(competencias(j)_i) ≠ categoria(competencias(j)_k) ;
  invariante competidoresInscriptos : (∀c ← competencias(j))incluida(participantes(c), atletas(j)) ;
  invariante jornadaValida : 1 ≤ jornadaActual(j) ≤ cantDias(j) ;
  invariante finalizadasSiiYaPasoElDia : lasPasadasFinalizaron(j) ∧ lasQueNoPasaronNoFinalizaron(j) ;
}
```

6. `problema dePaseo` (j: JJOO) = `result` : [Atleta]
   Devuelve los atletas que fueron a los juegos pero no participan en ninguna competencia.
7. `problema medallero` (j: JJOO) = `result` : [(Pais, [ℤ])]
   Devuelve el medallero. Para cada país que haya ganado al menos una medalla, debe haber una lista con la cantidad de medallas de oro, plata, y bronce que ganó respectivamente. El medallero debe estar ordenado de acuerdo a la cantidad de medallas de oro, plata y bronce de cada país. Un país que no obtuvo medalla en ninguna de las categorías no debe aparecer.
8. `problema boicotPorDisciplina` (j: JJOO, cat: (Deporte, Sexo), p: Pais) = `result` : ℤ
   Modifica los JJOO retirando a todos los atletas del pais p de la competencia de categoría cat, y devuelve la cantidad de atletas que se retiraron.
9. `problema losMasFracasados` (j: JJOO, p: Pais) = `result` : [Atleta]
   De todos atletas nacidos en el país p que en más competencias participaron, devuelve aquellos que no ganaron ninguna medalla.
10. `problema liuSong` (j: JJOO, a: Atleta, p: País)
    Modifica los juegos j, nacionalizando al atleta a al país p.
11. `problema stevenBradbury` (j: JJOO) = `result` : Atleta
    Devuelve al atleta con menos capacidad entre todos los ganadores de una medalla de oro.
12. `problema uyOrdenadoAsíHayUnPatrón` (j: JJOO) = `result` : Bool
    Dado un Juego Olímpico, indica si al tomar la lista de los mejores países en cada día(*) hasta la jornada actual, los países que estan en esa lista rotan de manera estricta.

(*) Por cada dia, el *mejor país* es aquél que más medallas de oro ganó. En caso de haber un empate, nos quedaremos con el menor país comparando lexicografimente. Si un día nadie ganó una medalla de oro, ese día se saltea y no aparece en la lista.

Ejemplo: La lista de los *mejores países* [Argentina, USA, Canada, Argentina, USA] rota estrictamente.
La lista de los *mejores países* [Argentina, USA, Canada, USA, Argentina] no rota estrictamente.

13. `problema sequíaOlímpica` (j: JJOO) = `result` : [País]
Dado un Juego Olímpico, devuelve todos los países participantes que más días pasaron sin ganar una medalla.

14. `problema transcurrirDia` (j: JJOO)
Modifica el juego reflejando que transcurrió un día. Los resultados de las competencias del día que pasó están dados por las capacidades de los atletas. En cada competencia se le realiza el control antidoping a un único atleta, elegido al azar, y el control le da positivo a lo sumo al 5 % de los atletas controlados.

5. Auxiliares

`aux ciaNumbers` (as: [Atleta]) : $[\mathbb{Z}] = [\, ciaNumber(a) \mid a \leftarrow as \,]$;
`aux competencias` (j: JJOO) : [Competencia] $= [\, c \mid d \leftarrow [1..cantDias(j)], c \leftarrow cronograma(j,d) \,]$;
`aux incluida` (l_1, l_2:[T]) : Bool $= (\forall x \leftarrow l_1) cuenta(x, l_1) \leq cuenta(x, l_2)$;
`aux lasPasadasFinalizaron` (j: JJOO) : Bool $= (\forall d \leftarrow [1..jornadaActual(j)))(\forall c \leftarrow cronograma(j,d)) finalizada(c)$;
`aux lasQueNoPasaronNoFinalizaron` (j: JJOO) : Bool $=$
$(\forall d \leftarrow (jornadaActual(j)..cantDias(j)])(\forall c \leftarrow cronograma(j,d)) \neg finalizada(c)$;
`aux ordenada` (l:[T]) : Bool $= (\forall i \leftarrow [0..|l|-1)) l_i \leq l_{i+1}$;
`aux sinRepetidos` (l: [T]) : Bool $= (\forall i, j \leftarrow [0..|l|), i \neq j) l_i \neq l_j$;