

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

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  $d \in \text{deportes}(a)$ ;  
  
  invariante  $|\text{deportes}(a)| > 0$ ;  
  invariante  $\text{sinRepetidos}(\text{deportes}(a))$ ;  
  invariante  $\text{ordenada}(\text{deportes}(a))$ ;  
  invariante  $\text{capacidadEnRango} : (\forall d \leftarrow \text{deportes}(a)) 0 \leq \text{capacidad}(a, d) \leq 100$ ;  
}  
  
problema especificidad (this: Atleta) = res : Deporte {  
  asegura  $res \in \text{deportes}(this)$ ;  
  asegura  $(\forall d \leftarrow \text{deportes}(this)) \text{capacidad}(this, res) \geq \text{capacidad}(this, d)$ ;  
}  
  
problema Atleta (this: Atleta, nom : String, s : Sexo, a :  $\mathbb{Z}$ , nac : Pais, cia :  $\mathbb{Z}$ ) {  
  modifica this;  
  asegura  $\text{nombre}(this) == nom$ ;  
  asegura  $\text{sexo}(this) == s$ ;  
  asegura  $\text{añoNacimiento}(this) == a$ ;  
  asegura  $\text{nacionalidad}(this) == nac$ ;  
  asegura  $\text{ciaNumber}(this) == cia$ ;  
  asegura  $\text{deportes}(this) == []$ ;  
}  
  
problema nombre (this : Atleta) = res : String {  
  asegura  $\text{nombre}(this) == result$ ;  
}  
  
problema sexo (this : Atleta) = res : Sexo {  
  asegura  $\text{sexo}(this) == result$ ;  
}  
  
problema añoNacimiento (this : Atleta) = res :  $\mathbb{Z}$  {  
  asegura  $\text{añoNacimiento}(this) == result$ ;  
}  
  
problema nacionalidad (this : Atleta) = res : Pais {  
  asegura  $\text{nacionalidad}(this) == result$ ;  
}
```

```

problema ciaNumber (this : Atleta) = res :  $\mathbb{Z}$  {
  asegura ciaNumber(this) == result;
}

problema deportes (this : Atleta) = res : [Deporte] {
  asegura deportes(this) == result;
}

problema capacidad (this : Atleta, d : Deporte) = res :  $\mathbb{Z}$  {
  requiere  $d \in \text{deportes}(this)$ ;
  asegura capacidad(this, d) == result;
}

problema entrenarNuevoDeporte (this: Atleta, d: Deporte, c:  $\mathbb{Z}$ ) {
  requiere  $0 \leq c \leq 100$ ;
  modifica this;
  asegura nombre(this) == nombre(pre(this));
  asegura sexo(result) == sexo(pre(this));
  asegura añoNacimiento(this) == añoNacimiento(pre(this));
  asegura nacionalidad(this) == nacionalidad(pre(this));
  asegura ciaNumber(this) == ciaNumber(pre(this));
  asegura mismos(deportes(this), sacarRepetidos(d : deportes(pre(this))));
  asegura  $(\forall x \leftarrow \text{deportes}(this), x \neq d) \text{capacidad}(this, x) == \text{capacidad}(\text{pre}(this), x)$ ;
  asegura capacidad(this, d) == c;
}

problema operator== (this, a: Atleta) = res : Bool {
  asegura result == mismosDatosPersonales(this, a)  $\wedge$  mismosDeportesYCapacidades(this, a);
  aux mismosDatosPersonales (a1, a2: Atleta) : Bool = nombre(a1) == nombre(a2)  $\wedge$  sexo(a1) == sexo(a2)
     $\wedge$  añoNacimiento(a1) == añoNacimiento(a2)  $\wedge$  nacionalidad(a1) == nacionalidad(a2)
     $\wedge$  ciaNumber(a1) == ciaNumber(a2);
  aux mismosDeportesYCapacidades (a1, a2: Atleta) : Bool =
    deportes(a1) == deportes(a2)  $\wedge$   $(\forall d \leftarrow \text{deportes}(a1)) \text{capacidad}(a1, d) == \text{capacidad}(a2, d)$ ;
}

```

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)  $\wedge$   $a \in \text{lesTocoControlAntiDoping}(c)$ ;

  invariante participaUnaSolaVez : sinRepetidos(ciaNumbers(participantes(c)));
  invariante participantesPertenecenACat :
     $(\forall p \leftarrow \text{participantes}(c)) \text{prm}(\text{categoria}(c)) \in \text{deportes}(p) \wedge \text{sgd}(\text{categoria}(c)) == \text{genero}(p)$ ;
  invariante elRankingEsDeParticipantesYNoHayRepetidos :
    finalizada(c)  $\Rightarrow$  incluida(ranking(c), participantes(c));
  invariante seControlanParticipantesYNoHayRepetidos :
    finalizada(c)  $\Rightarrow$  incluida(lesTocoControlAntiDoping(c), participantes(c));
}

problema Competencia (this: Competencia, d: Deporte, s: Sexo, as: [Atleta]) {
  requiere sonDeEstaCategoria :  $(\forall a \leftarrow as) d \in \text{deportes}(a) \wedge s == \text{sexo}(a)$ ;
  requiere noHayAtletasRepetidos : sinRepetidos(ciaNumbers(as));
  modifica this;
  asegura categoria(this) == (d, s);
  asegura mismos(participantes(this), as);
  asegura  $\neg \text{finalizada}(this)$ ;
}

```

```

}

problema categoria (this : Competencia) = res : (Deporte,Sexo) {
    asegura categoria(this) == result;
}

problema participantes (this : Competencia) = res : [Atleta] {
    asegura mismos(participantes(this),result);
}

problema finalizada (this : Competencia) = res : Bool {
    asegura finalizada(this) == result;
}

problema ranking (this : Competencia) = res : [Atleta] {
    requiere finalizada(this);
    asegura ranking(this) == result;
}

problema lesTocoControlAntiDoping (this : Competencia) = res : [Atleta] {
    requiere finalizada(this);
    asegura mismos(lesTocoControlAntiDoping(this),result);
}

problema leDioPositivo (this : Competencia, a : Atleta) = res : Bool {
    requiere finalizada(this);
    requiere a ∈ lesTocoControlAntiDoping(this);
    asegura leDioPositivo(this,a) == result;
}

problema finalizar (this: Competencia, posiciones: [Z], control: [(Z, Bool)]) {
    requiere ¬finalizada(this);
    requiere incluida(posiciones,ciaNumbers(participantes(this)));
    requiere incluida(primeros(control),ciaNumbers(participantes(this)));
    modifica this;
    asegura seMantieneCategoria : categoria(this) == categoria(pre(this));
    asegura seMantienenenParticipantes : mismos(participantes(this),participantes(pre(this)));
    asegura finalizo : finalizada(this);
    asegura rankingOrdenado : ciaNumbers(ranking(this)) == posiciones;
    asegura quienesSeControlan : mismos(ciaNumbers(lesTocoControlAntiDoping(this)),primeros(control));
    asegura resultadosDeControl : (∀x ← control) leDioPositivo(this,elAtleta(participantes(this),prm(x))) ⇔ sgd(x);
    aux elAtleta (as: [Atleta], x:Z) : Atleta = [a | a ← as, ciaNumber(a) == x]0;
}

problema linfordChristie (this: Competencia, ciaNum: Z) {
    requiere noFinalizada : ¬finalizada(this);
    requiere esParticipante : ciaNum ∈ ciaNumbers(participantes(this));
    modifica this;
    asegura seMantieneCategoria : categoria(this) == categoria(pre(this));
    asegura soloUnoDescalificado : mismos(participantes(this),[a | a ← participantes(pre(this)), ciaNumber(a) ≠ ciaNum]);
    asegura noFinalizada : ¬finalizada(this);
}

problema gananLosMasCapaces (this: Competencia) = res : Bool {
    requiere seConocenResultados : finalizada(this);
    asegura res == ordenada(reverso(capacidades(ranking(this),deporte(this))));
}

problema sancionarTramposos (this: Competencia) {
    requiere seConocenResultados : finalizada(this);
    modifica this;
    asegura seMantieneCategoria : categoria(this) == categoria(pre(this));
    asegura seMantienenParticipantes : mismos(participantes(this),participantes(pre(this)));
    asegura sigueFinalizada : finalizada(this);
    asegura drogadosDescalificados : ranking(this) == [a | a ← ranking(pre(this)), noLoDescubrenDopado(a,pre(this))]
}

```

```

asegura seMantieneControl : mismos(lesTocoControlAntiDoping(this), lesTocoControlAntiDoping(pre(this))) ;
asegura mismosResultadosControl :
  ( $\forall a \leftarrow lesTocoControlAntiDoping(this)$ )  $leDioPositivo(this, a) \Leftrightarrow leDioPositivo(pre(this), a)$  ;

aux noLoDescubrenDopado (a: Atleta, c: Competencia) : Bool =
   $a \notin lesTocoControlAntiDoping(c) \vee \neg leDioPositivo(c, a)$  ;
}

problema operator== (this, c: Competencia) = res : Bool {
  asegura mismosParticipantesYCategoria(this, c)  $\wedge$  finalizada(this) == finalizada(c)  $\wedge$ 
    (finalizada(this)  $\longrightarrow$  ranking(this) == ranking(c)  $\wedge$  coincidenControlesAntidoping(this, c)) ;

  aux mismosParticipantesYCategoria (c1, c2: Competencia) : Bool = mismos(participantes(c1), participantes(c2))  $\wedge$ 
    categoria(c1) == categoria(c2) ;
  aux coincidenControlesAntidoping (c1, c2: Competencia) : Bool =
    mismos(lesTocoControlAntidoping(c1), lesTocoControlAntidoping(c2))  $\wedge$ 
    ( $\forall a \leftarrow lesTocoControlAntidoping(c1)$ )  $leDioPositivo(c1, a) == leDioPositivo(c2, a)$  ;
}

```

4. JJOO

```

tipo JJOO {
  observador año (j: JJOO) :  $\mathbb{Z}$ ;
  observador atletas (j: JJOO) : [Atleta];
  observador cantDias (j: JJOO) :  $\mathbb{Z}$ ;
  observador cronograma (j: JJOO, dia:  $\mathbb{Z}$ ) : [Competencia];
    requiere  $1 \leq dia \leq cantDias(j)$ ;
  observador jornadaActual (j: JJOO) :  $\mathbb{Z}$ ;

  invariante atletasUnicos : sinRepetidos(ciaNumbers(atletas(j)));
  invariante unaDeCadaCategoria : ( $\forall i, k \leftarrow [0..|competencias(j)|], i \neq k$ )
    categoria(competencias(j)i)  $\neq$  categoria(competencias(j)k);
  invariante competidoresInscriptos : ( $\forall c \leftarrow competencias(j)$ ) incluida(participantes(c), atletas(j));
  invariante jornadaValida :  $1 \leq jornadaActual(j) \leq cantDias(j)$ ;
  invariante finalizadasSiiYaPasoElDia : lasPasadasFinalizaron(j)  $\wedge$  lasQueNoPasaronNoFinalizaron(j);
}

problema JJOO (this: JJOO, año:  $\mathbb{Z}$ , as: [Atleta], cron: [[Competencia]]) {
  requiere sinRepetidos(ciaNumbers(as));
  requiere ( $\forall cs \leftarrow concat(cron)$ ) ( $\neg(\exists i, j \leftarrow [0..|concat(cron)|], i \neq j)$  categoria(csi) == categoria(csj));
  requiere ( $\forall cs \leftarrow concat(cron)$ ) incluida(participantes(cs), as);
  requiere  $|cron| \geq 1$ ;
  requiere ( $\forall c \leftarrow concat(cron)$ )  $\neg finalizada(c)$ ;
  modifica this;
  asegura año == año(this);
  asegura mismos(atletas(this), as);
  asegura  $|cron| == cantDias(this)$ ;
  asegura ( $\forall j \leftarrow [0..|cron|]$ ) mismos(cronj, cronograma(this, j + 1));
  asegura jornadaActual(this) == 1;
}

problema año (this: JJOO) = res :  $\mathbb{Z}$  {
  asegura año(this) == result;
}

problema atletas (this: JJOO) = res : [Atleta] {
  asegura mismos(atletas(this), result);
}

problema cantDias (this: JJOO) = res :  $\mathbb{Z}$  {
  asegura cantDias(this) == result;
}

problema jornadaActual (this: JJOO) = res :  $\mathbb{Z}$  {
  asegura jornadaActual(this) == result;
}

```

```

}

problema cronograma (this: JJO, d:  $\mathbb{Z}$ ) = res : [Competencia] {
    requiere  $1 \leq d \leq \text{cantDias}(this)$ ;
    asegura  $\text{cronograma}(this, d) == \text{result}$ ;
}

problema competencias (this: JJO) = res : [Competencia] {
    asegura  $\text{result} == \text{competencias}(j)$ ;
}

problema competenciasFinalizadasConOroEnPodio (this: JJO) = res : [Competencia] {
    asegura  $\text{mismos}(\text{result}, [c | c \leftarrow \text{competencias}(j), \text{finalizada}(c) \wedge |\text{ranking}(c)| > 0])$ ;
}

problema dePaseo (this: JJO) = res : [Atleta] {
    asegura  $\text{noParticipanEnNinguna} : \text{mismos}(\text{res}, \text{fueronAPasear}(this))$ ;
    aux  $\text{fueronAPasear}(j: JJO) : [Atleta] = [a | a \leftarrow \text{atletas}(j), \neg(\exists c \leftarrow \text{competencias}(j)) a \in \text{participantes}(c)]$ ;
}

problema medallero (this: JJO) = res : [(Pais,  $\mathbb{Z}$ )] {
    asegura  $\text{paísesConMedallas} : \text{mismos}(\text{primeros}(\text{res}), \text{paísesQueGanaron}(this))$ ;
    asegura  $\text{cantidadMedallasCorrecta} : (\forall m \leftarrow \text{res}) |\text{sgd}(m)| == 3 \wedge$ 
         $\text{sgd}(m)_0 == |\text{filtrarPorPais}(\text{medallistasOro}(this), \text{prm}(m))| \wedge$ 
         $\text{sgd}(m)_1 == |\text{filtrarPorPais}(\text{medallistasPlata}(this), \text{prm}(m))| \wedge$ 
         $\text{sgd}(m)_2 == |\text{filtrarPorPais}(\text{medallistasBronce}(this), \text{prm}(m))|$ ;
    asegura  $\text{bienOrdenada} : (\forall i \leftarrow (0..|\text{res}|)) \text{masMedallas}(\text{sgd}(\text{res}_{i-1}), \text{sgd}(\text{res}_i))$ ;
    aux  $\text{paísesQueGanaron}(j: JJO) : [\text{Pais}] = \text{sacarRepetidos}(\text{nacionalidades}(\text{medallistasOro}(j) + +$ 
         $\text{medallistasPlata}(j) + + \text{medallistasBronce}(j)))$ ;
    aux  $\text{masMedallas}(x, y: [\mathbb{Z}]) : \text{Bool} = x_0 > y_0 \vee (x_0 == y_0 \wedge x_1 > y_1) \vee (x_0 == y_0 \wedge x_1 == y_1 \wedge x_2 \geq y_2)$ ;
}

problema boicotPorDisciplina (this: JJO, cat: (Deporte, Sexo), p: Pais) = res :  $\mathbb{Z}$  {
    requiere  $\text{esCategoriaValida} : (\exists c \leftarrow \text{competencias}(this)) \text{categoria}(c) == \text{cat}$ ;
    modifica  $this$ ;
    asegura  $\text{soloCambiaCronograma} : \text{año}(this) == \text{año}(\text{pre}(this)) \wedge \text{cantDias}(this) == \text{cantDias}(\text{pre}(this))$ 
         $\wedge \text{jornadaActual}(this) == \text{jornadaActual}(\text{pre}(this)) \wedge \text{mismos}(\text{atletas}(this), \text{atletas}(\text{pre}(this)))$ ;
    asegura  $\text{mismaCantDeCompetencias} : (\forall d \leftarrow [1..\text{cantDias}(this)]) |\text{cronograma}(this, d)| == |\text{cronograma}(\text{pre}(this), d)|$ ;
    asegura  $\text{lasOtrasCompetenciasNoCambian} : (\forall d \leftarrow [1..\text{cantDias}(this)]) (\forall c \leftarrow \text{cronograma}(\text{pre}(this), d), \text{categoria}(c) \neq$ 
         $\text{cat}) \text{laCompetenciaSeMantiene}(this, d, c)$ ;
    asegura  $\text{boicotAEsaCat} : (\exists c \leftarrow \text{cronograma}(this, \text{elDiaDeEsaCat}(\text{pre}(this), \text{cat})))$ 
         $\text{igualSalvoBoicot}(c, \text{competenciaDeCat}(\text{pre}(this), \text{cat}), p)$ ;
    asegura  $\text{result} == |\text{filtrarPorPais}(\text{participantes}(\text{competenciaDeCat}(\text{pre}(this), \text{cat})), p)|$ ;
    aux  $\text{elDiaDeEsaCat}(j: JJO, \text{cat}: (\text{Deporte}, \text{Sexo})) : \mathbb{Z} =$ 
         $[d | d \leftarrow [1..\text{cantDias}(j)], (\exists c \leftarrow \text{cronograma}(j, d)) \text{categoria}(c) == \text{cat}]_0$ ;
    aux  $\text{competenciaDeCat}(j: JJO, \text{cat}: (\text{Deporte}, \text{Sexo})) : \text{Competencia} = [c | c \leftarrow \text{competencias}(j), \text{categoria}(c) ==$ 
         $\text{cat}]_0$ ;
    aux  $\text{igualSalvoBoicot}(c, \text{prec}: \text{Competencia}, p: \text{Pais}) : \text{Bool} = \text{categoria}(c) == \text{categoria}(\text{prec}) \wedge$ 
         $\text{mismos}(\text{participantes}(c), \text{sacarLosDePais}(\text{participantes}(\text{prec}), p) \wedge \text{finalizada}(c) \Leftrightarrow \text{finalizada}(\text{prec})$ 
         $\wedge \text{finalizada}(c) \Rightarrow (\text{ranking}(c) == \text{sacarLosDePais}(\text{ranking}(\text{prec}), p) \wedge$ 
         $\text{mismos}(\text{lesTocoControlAntiDoping}(c), \text{sacarLosDePais}(\text{lesTocoControlAntiDoping}(\text{prec}), p))$ 
         $\wedge (\forall a \leftarrow \text{lesTocoControlAntiDoping}(c)) \text{leDioPositivo}(c, a) \Leftrightarrow \text{leDioPositivo}(\text{prec}, a))$ ;
    aux  $\text{sacarLosDePais}(as: [\text{Atleta}], p: \text{Pais}) : [\text{Atleta}] = [a | a \leftarrow as, \text{nacionalidad}(a) \neq p]$ ;
}

problema losMasFracasados (this: JJO, p: Pais) = res : [Atleta] {
    asegura  $\text{mismos}(\text{res}, \text{noGanaronMedallas}(this, \text{losMasParticipantes}(this, \text{atletasDelPais}(this, p))))$ ;
    aux  $\text{atletasDelPais}(j: JJO, p: \text{Pais}) : [\text{Atleta}] = [a | a \leftarrow \text{atletas}(j), \text{nacionalidad}(a) == p]$ ;
    aux  $\text{losMasParticipantes}(j: JJO, as: [\text{Atleta}]) : [\text{Atleta}] = [a | a \leftarrow as,$ 
         $(\forall x \leftarrow as) \text{cantCompetencias}(j, a) \geq \text{cantCompetencias}(j, x)]$ ;
    aux  $\text{cantCompetencias}(j: JJO, a: \text{Atleta}) : \mathbb{Z} = |[c | c \leftarrow \text{competencias}(j), a \in \text{participantes}(c)]|$ ;
    aux  $\text{noGanaronMedallas}(j: JJO, as: [\text{Atleta}]) : [\text{Atleta}] = [a | a \leftarrow as, \text{cantMedallas}(j, a) == 0]$ ;
    aux  $\text{cantMedallas}(j: JJO, a: \text{Atleta}) : \mathbb{Z} = |[c | c \leftarrow \text{competencias}(j), \text{estaEnElPodio}(c, a)]|$ ;
}

```

```

aux estaEnElPodio (c: Competencia, a: Atleta) : Bool = finalizada(c) ∧ (salioPrimero(c, a) ∨ salioSegundo(c, a) ∨
  salioTercero(c, a));
aux salioPrimero (c: Competencia, a: Atleta) : Bool = |ranking(c)| ≥ 1 ∧ ranking(c)0 == a;
aux salioSegundo (c: Competencia, a: Atleta) : Bool = |ranking(c)| ≥ 2 ∧ ranking(c)1 == a;
aux salioTercero (c: Competencia, a: Atleta) : Bool = |ranking(c)| ≥ 3 ∧ ranking(c)2 == a;
}

problema liuSong (this: JJOO, a: Atleta, p: País) {
  requiere estaLiu : a ∈ atletas(this);
  modifica this;
  asegura loDemasIgual : año(this) == año(pre(this)) ∧ cantDias(this) == cantDias(pre(this))
    ∧ jornadaActual(this) == jornadaActual(pre(this));
  asegura mismaCantidadAtletas : |atletas(this)| == |atletas(pre(this))|;
  asegura atletasIguales : (∀at1 ← atletas(pre(this)), ¬(at1 == a)) at1 ∈ atletas(this);
  asegura cambioLiu : (∀at1 ← atletas(pre(this)), at1 == a) (∃at2 ← atletas(this)) igualSalvoPais(at1, at2, p);
  asegura mismaCantDeCompetencias : (∀d ← [1..cantDias(this)]) |cronograma(pre(this), d)| == |cronograma(this, d)|;
  asegura lasOtrasCompetenciasNoCambian : (∀d ← [1..cantDias(this)]) (∀c ← cronograma(pre(this), d), a ∉ participantes
    laCompetenciaSeMantiene(this, d, c);
  asegura cambianLasDeLiu : (∀d ← [1..cantDias(this)]) (∀c ← cronograma(pre(this), d), a ∈ participantes(c))
    (∃c2 ← cronograma(this, d)) igualSalvoLiu(c, c2, a, p);

  aux igualSalvoPais (at1: Atleta, at2: Atleta, p: País) : Bool = nombre(at1) == nombre(at2) ∧
    sexo(at1) == sexo(at2) ∧ añoNacimiento(at1) == añoNacimiento(at2)
    ∧ ciaNumber(at1) == ciaNumber(at2) ∧ deportes(at1) == deportes(at2)
    ∧ (∀d ← deportes(at1)) capacidad(at1, d) == capacidad(at2, d) ∧ nacionalidad(at2) == p;
  aux igualSalvoLiu (c1: Competencia, c2: Competencia, a: Atleta, p: País) : Bool = categoria(c1) == categoria(c2)
    ∧ participantesYLiu(c1, c2, a, p) ∧ finalizada(c1) ⇔ finalizada(c2) ∧ finalizada(c1) ⇒
    (rankingYLiu(c1, c2, a, p) ∧ mismosControladosYLiu(c1, c2, a, p));
  aux participantesYLiu (c1: Competencia, c2: Competencia, a: Atleta, p: País) : Bool =
    |participantes(c1)| == |participantes(c2)|
    ∧ (∀at1 ← participantes(c1), at1! = a) at1 ∈ participantes(c2)
    ∧ (∀at1 ← participantes(c1), at1 == a) (∃at2 ← participantes(c2)) igualSalvoPais(at1, at2, p);
  aux rankingYLiu (c1: Competencia, c2: Competencia, a: Atleta, p: País) : Bool = |ranking(c1)| == |ranking(c2)|
    ∧ (∀i ← [0..|ranking(c1)|], ranking(c1)i! = a) ranking(c2)i == ranking(c1)i
    ∧ (∀i ← [0..|ranking(c1)|], ranking(c1)i == a) igualSalvoPais(ranking(c1)i, ranking(c2)i, p);
  aux mismosControladosYLiu (c1: Competencia, c2: Competencia, a: Atleta, p: País) : Bool =
    |lesTocoControlAntiDoping(c1)| == |lesTocoControlAntiDoping(c2)| ∧
    (∀at1 ← lesTocoControlAntiDoping(c1), at1! = a) at1 ∈ lesTocoControlAntiDoping(c2) ∧ leDioPositivo(c1, at1) ==
    leDioPositivo(c2, at1) ∧
    (∀at1 ← lesTocoControlAntiDoping(c1), at1 == a) (∃at2 ← lesTocoControlAntiDoping(c2))
    igualSalvoPais(at1, at2, p) ∧ leDioPositivo(c1, at1) == leDioPositivo(c2, at2);
}

problema stevenBradbury (this: JJOO) = res : Atleta {
  requiere alguienGanoMedalla : (∃d ← [1..jornadaActual(this)]) (∃c ← cronograma(this, d), finalizada(c)) |ranking(c)|
    0;
  asegura ganoMedallaDeOro : res ∈ primeros(ganadoresPorCategoria(this));
  asegura elMenosCapaz : (∀a ← primeros(ganadoresPorCategoria(this)))
    peorDesempeño(res, this) ≤ peorDesempeño(a, this);

  aux ganadoresPorCategoria (j: JJOO) : [(Atleta, (Deporte, Sexo))] = [(ranking(c)0, categoria(c)) |
    d ← [1..jornadaActual(j)], c ← cronograma(j, d), finalizada(c) ∧ |ranking(c)| > 0];
  aux peorDesempeño (a: Atleta, j: JJOO) : ℤ =
    minimo([capacidad(a, prm(sgd(g))) | g ← ganadoresPorCategoria(j), prm(g) == a]);
}

problema uyOrdenadoAsíHayUnPatrón (this: JJOO) = res : Bool {
  asegura siguenSiempreElMismoOrden(losMejoresPaises(this)) == res;

  aux losMejoresPaises (j: JJOO) : [País] = [mejorEseDia(j, i) | i ← [1..jornadaActual(j)], alguienGanoOro(j, i)];
  aux mejorEseDia (j: JJOO, d: ℤ) : País = [p | p ← paises(j),
    ¬(∃p2 ← paises(j)) (cantOro(j, p2, d) > cantOro(j, p, d) ∨ (cantOro(j, p2, d) == cantOro(j, p, d) ∧ p2 < p))] ]0;
  aux cantOro (j: JJOO, p: País, d: ℤ) : ℤ = |[1 | c ← cronograma(j, d), finalizada(c)
    ∧ ranking(c) ≥ 1 ∧ nacionalidad(ranking(c)0) == p]|;
  aux alguienGanoOro (j: JJOO, d: ℤ) : Bool = (∃c ← cronograma(j, d)) finalizada(c) ∧ ranking(c) ≥ 1;
}

```

```

    aux siguenSiempreElMismoOrden (ps:[Pais]) : Bool = ( $\forall i, j \leftarrow [0..|ps| - 1], i < j \wedge ps_i == ps_j) ps_{i+1} == ps_{j+1} \wedge$ 
      ( $\forall i, j \leftarrow [1..|ps|], i < j \wedge ps_i == ps_j) ps_{i-1} == ps_{j-1}$  ;
  }

problema sequiaOlimpica (this: JJOO) = res : [País] {
  asegura mismos(result, secosOlimpicos(this));

  aux secosOlimpicos (j: JJOO) : [País] = [p | p  $\leftarrow$  paises(j), masDiasSinMedallas(j, p) == maxDiasSinMedallas(j)];
  aux masDiasSinMedallas (j: JJOO, p: País) :  $\mathbb{Z}$  = maxDif(0 : [i | i  $\leftarrow$  [1..jornadaActual(j)],
    GanoMedallaEseDia(j, p, i)] ++ [jornadaActual(j)]);
  aux maxDif (ls:[ $\mathbb{Z}$ ]) :  $\mathbb{Z}$  = max([lsi - lsi-1 | i  $\leftarrow$  [1..|ls|]]);
  aux GanoMedallaEseDia (j: JJOO, p: País, i:  $\mathbb{Z}$ ) : Bool = ( $\exists c \leftarrow$  cronograma(j, i))
    (|ranking(c)|  $\geq$  1  $\wedge$  nacionalidad(ranking(c)0) == p)
     $\vee$  (|ranking(c)|  $\geq$  2  $\wedge$  nacionalidad(ranking(c)1) == p)
     $\vee$  (|ranking(c)|  $\geq$  3  $\wedge$  nacionalidad(ranking(c)2) == p);
  aux maxDiasSinMedallas (j: JJOO) :  $\mathbb{Z}$  = max([masDiasSinMedallas(j, p) | p  $\leftarrow$  paises(j)]);
}

problema transcurrirDia (this: JJOO) {
  requiere losJuegosNoTerminaron : jornadaActual(this)  $\leq$  cantDias(this);
  modifica this;
  asegura seMantieneAño : año(this) == año(pre(this));
  asegura seMantienenAtletas : mismos(atletas(this), atletas(pre(this)));
  asegura seMantienenDias : cantDias(this) == cantDias(pre(this));
  asegura avanzaDia : jornadaActual(this) == jornadaActual(pre(this)) + 1;
  asegura mismaCantDeCompetencias : ( $\forall d \leftarrow$  [1..cantDias(this)]) |cronograma(this, d)| == |cronograma(pre(this), d)|;
  asegura cronogramaDeOtrosDiasNoCambia : ( $\forall d \leftarrow$  [1..cantDias(this)], d  $\neq$  jornadaActual(pre(this)))
    ( $\forall c \leftarrow$  cronograma(pre(this), d)) laCompetenciaSeMantiene(this, d, c);
  asegura lasFinalizadasSeMantienen : ( $\forall c \leftarrow$  cronograma(pre(this), jornadaActual(pre(this))), finalizada(c))
    laCompetenciaSeMantiene(this, jornadaActual(pre(this)), c);
  asegura finalizanCompetencias : ( $\forall c \leftarrow$  cronograma(pre(this), jornadaActual(pre(this))),  $\neg$ finalizada(c))
    finaliza(this, c, jornadaActual(pre(this)));

  aux finaliza (j: JJOO, c: Competencia, dia:  $\mathbb{Z}$ ) : Bool = ( $\exists x \leftarrow$  cronograma(j, dia)) categoria(x) == categoria(c)  $\wedge$ 
    mismos(participantes(x), participantes(c))  $\wedge$  finalizada(x)  $\wedge$  mismos(ranking(x), participantes(x))  $\wedge$ 
    ordenada(reverso(capacidades(ranking(x), deporte(x))))  $\wedge$ 
    |ranking(x)|  $\geq$  1  $\Rightarrow$  |lesTocoControlAntiDoping(x)| == 1;
}

problema operator== (this, j: JJOO) = res : Bool {
  asegura result == (año(this) == año(j)  $\wedge$  cantDias(this) == cantDias(j)
     $\wedge$  jornadaActual(this) == jornadaActual(j)  $\wedge$  mismos(atletas(this), atletas(j))  $\wedge$  mismoCronograma(this, j));
  aux mismoCronograma (j1, j2: JJOO) : Bool = ( $\forall d \leftarrow$  [1..cantDias(j1)]) mismos(cronograma(j1, d), cronograma(j2, d));
}

```

5. Auxiliares

```

  aux ciaNumbers (as: [Atleta]) : [ $\mathbb{Z}$ ] = [ciaNumber(a) | a  $\leftarrow$  as];
  aux competencias (j: JJOO) : [Competencia] = [c | d  $\leftarrow$  [1..cantDias(j)], c  $\leftarrow$  cronograma(j, d)];
  aux incluida (l1, l2: [T]) : Bool = ( $\forall x \leftarrow$  l1) cuenta(x, l1)  $\leq$  cuenta(x, l2);
  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]) li  $\leq$  li+1;
  aux sinRepetidos (l: [T]) : Bool = ( $\forall i, j \leftarrow$  [0..|l|], i  $\neq$  j) li  $\neq$  lj;
  aux capacidades (as: [Atleta], d: Deporte) : [ $\mathbb{Z}$ ] = [capacidad(a, d) | a  $\leftarrow$  as];
  aux deporte (c: Competencia) : Deporte = prm(categoria(c));
  aux laCompetenciaSeMantiene (j: JJOO, d:  $\mathbb{Z}$ , c: Competencia) : Bool =
    ( $\exists x \leftarrow$  cronograma(j, d)) categoria(x) == categoria(c)  $\wedge$  mismos(participantes(x), participantes(c))
     $\wedge$  finalizada(x)  $\Leftrightarrow$  finalizada(c)  $\wedge$  finalizada(x)  $\Rightarrow$  (ranking(x) == ranking(c)  $\wedge$  mismosControlados(x, c);
    aux medallistasOro (j: JJOO) : [Atleta] = [ranking(c)0 | d  $\leftarrow$  [1..jornadaActual(j)], c  $\leftarrow$  cronograma(j, d),
    finalizada(c)  $\wedge$  |ranking(c)|  $\geq$  1];
    aux medallistasPlata (j: JJOO) : [Atleta] = [ranking(c)1 | d  $\leftarrow$  [1..jornadaActual(j)], c  $\leftarrow$  cronograma(j, d),
    finalizada(c)  $\wedge$  |ranking(c)|  $\geq$  2];

```

```

    aux medallistasBronce (j: JJOO) : [Atleta] = [ranking(c)2 | d ← [1..jornadaActual(j)], c ← cronograma(j, d),
finalizada(c) ∧ |ranking(c)| ≥ 3];
    aux minimo (l: [Z]) : Z = [x | x ← l, (∀y ← l) x ≤ y]0;
    aux mismosControlados (c1, c2: Competencia) : Bool =
mismos(lesTocoControlAntiDoping(c1), lesTocoControlAntiDoping(c2)) ∧
(∀p ← lesTocoControlAntiDoping(c1)) leDioPositivo(c1, p) ⇔ leDioPositivo(c2, p);
    aux nacionalidades (as: [Atleta]) : [Pais] = [nacionalidad(a) | a ← as];
    aux primeros (l: [(T,S)]) : [T] = [prm(x) | x ← l];
    aux reverso (l: [T]) : [T] = [l|x|-i-1 | i ← [0..|l|)];
    aux sacarRepetidos (l: [T]) : [T] = [li | i ← [0..|l|], li ∉ l[0..i)];

```