

TPE 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 especialidad (a: Atleta) = res : Deporte {  
  asegura  $\text{res} \in \text{deportes}(a)$ ;  
  asegura  $\text{capacidad}(a, \text{res}) == \text{maximoLista}(\text{listarCapacidades}(a))$ ;  
  aux listarCapacidades (a: Atleta) : [ $\mathbb{Z}$ ] =  $[\text{capacidad}(a, x) | x \leftarrow \text{deportes}(a)]$ ;  
  aux maximoLista (ls: [ $\mathbb{Z}$ ]) :  $\mathbb{Z}$  =  $[x | x \leftarrow \text{ls}, (\forall y \leftarrow \text{ls}) x \geq y][0]$ ;  
}
```

```
problema entrenarNuevoDeporte (a: Atleta, d: Deporte, c:  $\mathbb{Z}$ ) {  
  requiere  $c \geq 0 \wedge c \leq 100$ ;  
  requiere  $d \notin \text{deportes}(a)$ ;  
  modifica  $a$ ;  
  asegura  $\text{nombre}(a) == \text{nombre}(\text{pre}(a))$ ;  
  asegura  $\text{sexo}(a) == \text{sexo}(\text{pre}(a))$ ;  
  asegura  $\text{añoNacimiento}(a) == \text{añoNacimiento}(\text{pre}(a))$ ;  
  asegura  $\text{nacionalidad}(a) == \text{nacionalidad}(\text{pre}(a))$ ;  
  asegura  $\text{ciaNumber}(a) == \text{ciaNumber}(\text{pre}(a))$ ;  
  asegura  $(\forall d' \leftarrow \text{deportes}(\text{pre}(a))) \text{capacidad}(a, d') == \text{capacidad}(\text{pre}(a), d')$ ;  
  asegura  $\text{mismos}(\text{deportes}(a), d : \text{deportes}(\text{pre}(a)))$ ;  
  asegura  $\text{ordenada}(\text{deportes}(a))$ ;  
  asegura  $\text{capacidad}(a, d) == c$ ;  
}
```

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 participantesPerteneceenACat :
   $(\forall p \leftarrow participantes(c)) prm(categoria(c)) \in deportes(p) \wedge sgd(categoria(c)) == sexo(p)$ ;
invariante elRankingEsDeParticipantesYNoHayRepetidos :
   $finalizada(c) \Rightarrow incluida(ranking(c), participantes(c))$ ;
invariante seControlanParticipantesYNoHayRepetidos :
   $finalizada(c) \Rightarrow incluida(lesTocoControlAntiDoping(c), participantes(c))$ ;
}

problema finalizarCompetencia (c: Competencia, posiciones: [Atleta], control: [(Atleta, Bool)]) {
  requiere  $\neg finalizada(c)$ ;
  requiere contenida(posiciones, participantes(c));
  requiere  $sinRepetidos(posiciones)$ ;
  requiere  $(\forall x \leftarrow control) prm(x) \in participantes(c)$ ;
  requiere  $sinRepetidos([prm(x)|x \leftarrow control])$ ;
  modifica c;
  asegura finalizada(c);
  asegura  $ranking(c) == posiciones$ ;
  asegura  $participantes(c) == participantes(pre(c))$ ;
  asegura  $mismos(lesTocoControlAntiDoping(c), [prm(x)|x \leftarrow control])$ ;
  asegura  $(\forall x \leftarrow control) leDioPositivo(prm(x)) == sgn(x)$ ;
  asegura  $categoria(c) == categoria(pre(c))$ ;
}

problema linfordChristie (c: Competencia, a: Atleta) {
  requiere  $\neg finalizada(c)$ ;
  requiere  $a \in participantes(c)$ ;
  modifica c;
  asegura  $mismos(participantes(pre(c)), a : participantes(c))$ ;
  asegura  $categoria(c) == categoria(pre(c))$ ;
  asegura  $\neg finalizada(c)$ ;
}

problema gananLosMasCapaces (c: Competencia) = res : Bool {
  requiere finalizada(c);
  asegura  $res == (\forall x \leftarrow [0..|ranking(c)| - 2])$ 
     $capacidad(ranking(c)[x], prm(categoria(c))) \geq capacidad(ranking(c)[x + 1], prm(categoria(c)))$ ;
}

problema sancionarTramposos (c: Competencia) {
  requiere finalizada(c);
  modifica c;
  asegura finalizada(c);
  asegura  $categoria(c) == categoria(pre(c))$ ;
  asegura  $mismos(participantes(c), participantes(pre(c)))$ ;
  asegura  $incluida(ranking(c), participantes(c))$ ;
  asegura  $incluida(lesTocoControlAntiDoping(c), participantes(c))$ ;
  asegura  $mismos(ranking(pre(c)), ranking(c) ++ listaDoping(c, lesTocoControlAntiDoping(c)))$ ;
  asegura  $ranking == [el|el \leftarrow ranking(pre(c)), el \notin listaDoping(c, lesTocoControlAntiDoping(c))]$ ;
  aux listaDoping (c:competencia, a:[atleta]) : [atleta] =  $[x|x \leftarrow a, leDioPositivo(c, x)]$ ;
}

```

4. JJOO

```

tipo JJOO {
  observador año (j: JJOO) :  $\mathbb{Z}$ ;

```

```

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

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 dePaseo (j: JJO) = res : [Atleta] {
    asegura mismos(res, noParticiparon(j));
    aux noParticiparon (j: JJO) : [Atleta] =  $[x | x \leftarrow atletas(j), \neg en(x, participaronConRepes(j))]$ ;
    aux participaronConRepes (j: JJO) :
        [Atleta] =  $[atl | dia \leftarrow [1..cantDias(j)], comp \leftarrow cronograma(j, dia), atl \leftarrow atletas(j), en(atl, participantes(comp))]$ ;
}

problema medallero (j: JJO) = res : [(Pais,  $\mathbb{Z}$ )] {
    asegura mismos(result, sacarSinMedallas(crearMedallero(j)));
    asegura medalleroOrdenado(result);
    aux listaNacionalidadesAtletas (j: JJO) : [Pais] =  $[nacionalidad(at) | at \leftarrow atletas(j)]$ ;
    aux sacarRepeticiones (ls : [T]) : [T] =  $[ls[x] | x \leftarrow [0..|ls| - 1], \neg en(ls[x], ls[0..x - 1])]$ ;
    aux listaPaises (j: JJO) : [Pais] = sacarRepeticiones(listaNacionalidadesAtletas(j));
    aux cuenta (x: T, a: [T]) :  $\mathbb{Z}$  =  $[y | y \leftarrow a, y == x]$ ;
    aux nacionalidadPrimerosTres (c: Competencia) : [Pais] =
        ifthenelse( $i \leq |ranking(c)| - 1$ , nacionalidad(ranking(c)[i]), "Narnia") |  $i \leftarrow [0, 2]$ ;
    aux ganadoresMTotales (M:  $\mathbb{Z}$ , j: JJO) : [Pais] =
         $[nacionalidadPrimerosTres(c)[M] | dia \leftarrow [1..jornadaActual(j)], c \leftarrow cronograma(j, dia), finalizada(c)]$ ;
    aux crearMedallero (j: JJO) : [(Pais,  $\mathbb{Z}$ )] =
         $[(p, [cuenta(p, ganadoresMTotales(0, j)), cuenta(p, ganadoresMTotales(1, j)),$ 
         $cuenta(p, ganadoresMTotales(2, j))]) | p \leftarrow listaPaises(j)]$ ;
    aux sacarSinMedallas (lista : [(Pais,  $\mathbb{Z}$ )] : [(Pais,  $\mathbb{Z}$ )] =  $[x | x \leftarrow lista, sgn(x) \neq [0, 0, 0]]$ ;
    aux medalleroOrdenado (lista : [(Pais,  $\mathbb{Z}$ )] : Bool =
         $(\forall x \leftarrow [0..|lista| - 2])((sgd(lista[x])[0] \geq sgd(lista[x + 1])[0])$ 
         $\vee ((sgd(lista[x])[0] == sgd(lista[x + 1])[0]) \wedge (sgd(lista[x])[1] \geq sgd(lista[x + 1])[1]))$ 
         $\vee ((sgd(lista[x])[0] == sgd(lista[x + 1])[0]) \wedge (sgd(lista[x])[1] == sgd(lista[x + 1])[1])$ 
         $\wedge (sgd(lista[x])[2] \geq sgd(lista[x + 1])[2])))$ ;
}

problema boicotPorDisciplina (j: JJO, cat: (Deporte, Sexo), p: Pais) = res :  $\mathbb{Z}$  {
    requiere existe( $c \leftarrow todasLasCompetencias(j)$ ) categoria(c) == cat;
    requiere finalizada(competenciaFiltrada(j, cat)) == false;
    modifica j;
    asegura finalizada(competenciaFiltrada(j, cat)) == finalizada(competenciaFiltrada(pre(j), cat));
    asegura año(j) == año(pre(j));
    asegura mismos(atletas(j), atletas(pre(j)));
    asegura cantDias(j) == cantDias(pre(j));
    asegura jornadaActual(j) == jornadaActual(pre(j));
    asegura  $(\forall dia \leftarrow [1..cantDias(j)])$ 
         $[cronograma(j, dia)] == [cronograma(pre(j), dia)]$ ;
    asegura  $(\forall dia \leftarrow [1..cantDias(j)]) (\forall x \leftarrow [1..(long(cronograma(j, dia)) - 1)])$ 
         $((cronograma(j, dia)[x] == cronograma(pre(j), dia)[x])$ 
         $\vee (categoria(cronograma(j, dia)[x]) == (categoria(cronograma(pre(j), dia)[x]))$ 
         $\wedge (categoria(cronograma(j, dia)[x]) == cat)))$ ;
    asegura  $(\forall comp \leftarrow todasLasCompetencias(j), categoria(comp) == cat)$ 
         $(\forall (atl \leftarrow participantes(comp)) nacionalidad(atl) \neq p)$ ;
    asegura mismos(participantes(competenciaFiltrada(j, cat)),
         $[x | x \leftarrow participantes(competenciaFiltrada(pre(j), cat)), nacionalidad(x) \neq p]$ );
    asegura res ==  $[x | comp \leftarrow todasLasCompetencias(pre(j)),$ 
         $x \leftarrow participantes(comp), (categoria(comp) == cat \wedge nacionalidad(x) == p)]$ ;
}

```

```

    aux todasLasCompetencias (j:JJOO) : [competencia] = [x|y ← [1..cantDIas(j)], x ← cronogramas(j,y)];
    aux competenciaFiltrada (j:JJOO, c:(deporte,sexo)) : competencia =
        cab([x|x ← todasLasCompetencias(j), c == categoria(x)]);
}

problema losMasFracasados (j: JJOO, p: Pais) = res : [Atleta] {
    asegura noGanaronMedallas : (∀x ← [noGanoMedallas(atl, comp)
        |atl ← result, comp ← todasLasCompetencias(j), finalizada(comp)])x;
    asegura todosDelPais : (∀x ← result)nacionalidad(x) == p;
    asegura (∀x ← result, y ← atletas(j), nacionalidad(y) == p)
        competenciasEnLasQueParticipo(j, x) >= competenciasEnLasQueParticipo(j, y);
    aux competenciasEnLasQueParticipo (j: JJOO, a:Atleta) : ℤ =
        long([z|z ← todasLasCompetencias(j), en(a, participantes(z))]);
    aux noGanoMedallas (atl: Atleta, comp: Competencia) : Bool =
        ciaNumber(ranking(comp)[0]) ≠ ciaNumber(atl) ∧ ciaNumber(ranking(comp)[1]) ≠ ciaNumber(atl)
        ∧ ciaNumber(ranking(comp)[2]) ≠ ciaNumber(atl);
}

problema liuSong (j: JJOO, a: Atleta, p: País) {
    requiere a ∈ atletas(j);
    modifica j;
    asegura año(j) == año(pre(j));
    asegura (∀i ← [0..|atletas(pre(j))| - 1], atletas(pre(j))i! = a)atletas(pre(j))i == atletas(j)i;
    asegura (∀x ← atletas(j), ciaNumber(x) == ciaNumber(a))
        (nombre(x) == nombre(a) ∧ sexo(x) == sexo(a) ∧ añoNacimiento(x) == añoNacimiento(a)
        ∧ nacionalidad(x) == p ∧ deportes(x) == deportes(a) ∧ capacidad(x) == capacidad(a));
    asegura cantDias(j) == cantDias(pre(j));
    asegura jornadaActual(j) == jornadaActual(pre(j));
    asegura (∀d ← [1..cantDias(j)]|cronograma(d, j)| == |cronograma(d, pre(j))|
        ∧ (∀i ← [0..|cronograma(d, j)| - 1])
        categoria(cronograma(d, j)[i]) == categoria(cronograma(d, pre(j))[i])
        ∧ finalizada(cronograma(d, j)[i]) == finalizada(cronograma(d, pre(j))[i]);
    asegura todosIgualesMenosLiuSong(partComp(j), partComp(pre(j)));
    asegura todoIgualMenosNacionalidad(partComp(j));
    asegura todosIgualesMenosLiuSong(rankComp(j), rankComp(pre(j)));
    asegura todoIgualMenosNacionalidad(rankComp(j));
    asegura todosIgualesMenosLiuSong(doppComp(j), doppComp(pre(j)));
    asegura todoIgualMenosNacionalidad(doppComp(j));
    asegura ∀(dia ← [1..jornadaActual(j)], comp ← [0..|cronograma(j, dia) - 1|], finalizada(cronograma(j, dia)))
        leDioPositivo(cronograma(j, dia)[comp], atl) == leDioPositivo(cronograma(pre(j), dia)[comp], atl);
    aux rankComp (j: JJOO) : [Atleta] =
        [atl|dia ← [1..cantDias(j)], comp ← cronograma(j, dia), atl ← ranking(comp), finalizada(comp)];
    aux doppComp (j: JJOO) : [Atleta] =
        [atl|dia ← [1..cantDias(j)], comp ← cronograma(j, dia),
        atl ← lesTocoControlAntiDopping(comp), finalizada(comp)];
    aux partComp (j: JJOO) : [Atleta] =
        [atl|dia ← [1..cantDias(j)], comp ← cronograma(j, dia), atl ← participantes(comp)];
    aux todosIgualesMenosLiuSong (ls : [Atleta], lsOld : [Atleta]) : Bool =
        ∀(x ← [0..|ls| - 1])(ls[x] == lsOld[x] ∨ ciaNumber(ls) == ciaNumber(a));
    aux todoIgualMenosNacionalidad (ls: [Atleta]) : Bool =
        ∀(x ← ls, ciaNumber(x) == ciaNumber(a))(nombre(x) == nombre(a) ∧ sexo(x) == sexo(a)
        ∧ añoNacimiento(x) == añoNacimiento(a) ∧ nacionalidad(x) == p ∧ deportes(x) == deportes(a)
        ∧ capacidad(x) == capacidad(a));
}

problema stevenBradbury (j: JJOO) = res : Atleta {
    requiere algunOro :
        long([comp|comp ← todasLasCompetencias(j), finalizada(comp) ∧ |ranking(comp)| > 0]) > 0;
    asegura ganoOro :
        existe(f ← [ranking(comp)[0]|comp ← todasLasCompetencias(j),
        finalizada(comp) ∧ |ranking(comp)| > 0])ciaNumber(result) == ciaNumber(f);
    asegura esElPeor :
        existe(dep ← deportesEnLosQueGano(j, result))

```

```

    paraTodo( $f \leftarrow [\text{ranking}(\text{comp})[0] | \text{comp} \leftarrow \text{todasLasCompetencias}(j),$ 
       $\text{finalizada}(\text{comp}) \wedge |\text{ranking}(\text{comp})| > 0$ ])  $\text{capacidad}(\text{result}, \text{dep}) \leq \text{capacidad}(f, \text{prm}(\text{categoria}(\text{comp})))$ ;
  aux deportesEnLosQueGanoOro ( $j$ : JJOO,  $a$ : Atleta) : [Deportes] =
     $[\text{prm}(\text{categoria}(\text{comp})) | \text{comp} \leftarrow \text{todasLasCompetencias}(j), \text{finalizada}(\text{comp})$ 
       $\wedge |\text{ranking}(\text{comp})| > 0 \wedge \text{ciaNumber}(\text{ranking}(\text{comp})[0]) == \text{ciaNumber}(a)]$ ;
}

problema uyOrdenadoAsíHayUnPatrón ( $j$ : JJOO) = res : Bool {
  asegura res ==  $\text{ifthenelse}(|\text{mejoresPaises}(j)| == |\text{sinRepes}(\text{mejoresPaises}(j))|, \text{true},$ 
     $\text{ifthenelse}(\text{contar}(\text{cab}(\text{mejoresPaises}(j))) == 1 \wedge |\text{mejoresPaises}(j)| > |\text{sinRepes}(\text{mejoresPaises}(j))|, \text{false},$ 
       $(\forall i \leftarrow [0..|\text{mejoresPaises}(j)| - 1], k \leftarrow [0..|\text{mejoresPaises}(j)| - 1], i \bmod$ 
         $|\text{patron}(\text{mejoresPaises}(j))| == k \bmod |\text{patron}(\text{mejoresPaises}(j))|) s[i] == s[k]$ ;
  aux mejoresPaises ( $j$ : JJOO) : [Pais] =  $[\text{cab}(\text{ordenar}(\text{masOrosDelDia}(j, d))$ 
     $|d \leftarrow [1..|\text{jornadaActual}(j)| - 1], \text{cronograma}(j, d) \neq [], \text{masOrosDelDia}(j, d) \neq []]$ ;
  aux patron (( $ls$ : [T])) : [T] =
     $\text{ifthenelse}(|ls| == 1, ls, \text{sub}(ls, 0, \min(|i| \leftarrow [1..|ls| - 1], ls[i] == ls[0])))$ ;
  aux masOrosDelDia ( $j$ : JJOO,  $d$ : dia) : [Pais] =
     $\text{filtrarTuplaPorPrmMax}(j, \text{uniqSgdLista}(\text{tuplasMedallOroComaPais}(j, d)), d)$ ;
  aux tuplasMedallaOroComaPais ( $j$ : JJOO,  $d$ : Dia) : [(Z, Pais)] =
     $[(\text{contar}(\text{pais}, \text{PaisesConOroTalDia}(j, d)), p) | p \leftarrow \text{paisesConOroTalDia}(j, d)]$ ;
  aux paisesConOroTalDia ( $j$ : JJOO,  $d$ : Dia) : [Pais] =
     $[\text{pais}(\text{cab}(\text{ranking}(c)) | c \leftarrow \text{cronograma}(j, d), \text{finalizada}(c) \wedge |\text{ranking}(c)| > 0)]$ ;
  aux uniqSgdLista ( $ls$ : [(Z, Z)]) : [(Z, Z)] =
     $[ls(i) | i \leftarrow [0..|ls| - 1], j \leftarrow [0..|ls| - 1], i \neq j, \text{sgd}(ls[i]) \neq \text{sgd}(ls[j])]$ ;
  aux filtrarTuplaPorPrmMax ( $j$ : JJOO,  $ls$ : [(Z, Pais)],  $d$ : Dia) : [Pais] =
     $[\text{sgd}(tupla) | tupla \leftarrow \text{tuplasMedallaOroComaPais}(j, d), \text{prm}(tupla) \geq$ 
       $\text{maxListaTuplas}(\text{tuplasMedallaOroComaPais}(j, d))]$ ;
  aux maxListaTuplas ( $ls$ : [(ent, Pais)]) : Z =  $\text{max}([\text{prm}(tupla) | tupla \leftarrow ls])$ ;
}

problema sequiaOlimpica ( $j$ : JJOO) = res : [País] {
  asegura ( $\forall p \leftarrow \text{result}$ )  $\text{sequiaPais}(j, p) == \text{maximaSequia}(j)$ ;
  asegura ( $\forall p \leftarrow \text{listaPaisesConSequia}(j), \text{sequiaPais}(j, p) == \text{maximaSequia}(j)$ )  $p \in \text{result}$ ;
  aux sequiaPais ( $j$ : JJOO,  $p$ : pais) : Z =
     $\text{cerosSeguidos}([\text{ganoEnElDia}(j, \text{dia}, p) | \text{dia} \leftarrow [1..|\text{cantDias}(j)|]])$ ;
  aux ganoEnElDia ( $j$ : JJOO,  $\text{dia}$ : Z,  $\text{pais}$ : Pais) : Bool =
     $\text{long}([1 | c \leftarrow \text{cronograma}(j, \text{dia}), \text{en}(\text{pais}, \text{paisesQueGanaronCompetencia}(c))]) > 0$ ;
  aux listaPaisesConSequia ( $j$ : JJOO) : [(Pais, Int)] =
     $[(p, \text{sequiaPais}(j, p)) | p \leftarrow \text{listaPaises}(j)]$ ;
  aux maximaSequia ( $j$ : JJOO) : Z =
     $\text{maximoLista}([\text{sequiaPais}(j, p) | p \leftarrow \text{listaPaises}(j)])$ ;
  aux paisesQueGanaronCompetencia ( $c$ : Competencia) : [paises] =
     $[\text{pais}(\text{ranking}(c)[a]) | a \leftarrow [0..|\text{ranking}(c)| - 1], a \leq 2]$ ;
  aux generoTodosLosParesCeroComaUno ( $ls$ : [Bool]) : [(Z, Z)] =
     $[(c, c1) | c \leftarrow [0..|ls| - 1], c1 \leftarrow [c..|ls| - 1], ls[c] == \text{False} \wedge ls[c1] == \text{True}]$ ;
  aux uniqPrmLista ( $ls$ : [(Z, Z)]) : [(Z, Z)] =
     $[ls(i) | i \leftarrow [0..|ls| - 1], j \leftarrow [0..|ls| - 1], i \neq j, \text{prm}(ls[i]) \neq \text{prm}(ls[j])]$ ;
  aux uniqSgdLista ( $ls$ : [(Z, Z)]) : [(Z, Z)] =
     $[ls(i) | i \leftarrow [0..|ls| - 1], j \leftarrow [0..|ls| - 1], i \neq j, \text{sgd}(ls[i]) \neq \text{sgd}(ls[j])]$ ;
  aux diferenciaPrmSgd ( $ls$ : [(Z, Z)]) : [Z] =  $[\text{sgd}(el) - \text{prm}(el) | el \leftarrow ls]$ ;
  aux cerosSeguidos ( $ls$ : [Bool]) : Z =
     $\text{max}(\text{diferenciaPrmSgd}(\text{uniqSgdLista}(\text{uniqPrmLista}(\text{generoTodosLosParesCeroComaUno}(ls)))))$ ;
}

problema transcurrirDia ( $j$ : JJOO) {
  modifica  $j$ ;
  asegura  $\text{cantDias}(j) == \text{cantDias}(\text{pre}(j))$ ;
  asegura  $\text{jornadaActual}(j) == \text{jornadaActual}(\text{pre}(j)) + 1$ ;
  asegura  $\text{año}(j) == \text{año}(\text{pre}(j))$ ;
  asegura  $\text{mismos}(\text{atletas}(j), \text{atletas}(\text{pre}(j)))$ ;
  asegura ( $\forall \text{dia} \leftarrow [1..|\text{jornadaActual}(j)| - 1]$ )
     $\text{mismos}(\text{cronograma}(j, \text{dia}), \text{cronograma}(\text{pre}(j), \text{dia}))$ ;
  asegura  $|\text{cronograma}(j, \text{jornadaActual}(j))| == |\text{cronograma}(\text{pre}(j), \text{jornadaActual}(j))|$ ;
}

```

```

asegura ( $\forall i \leftarrow [0, |cronograma(j, jornadaActual(j))| - 1]$ )
  categoria(cronograma(j, jornadaActual(j))[i]) == categoria(cronograma(pre(j), jornadaActual(j))[i]);
asegura ( $\forall i \leftarrow [0, |cronograma(j, jornadaActual(j))| - 1]$ )
  mismos(participantes(cronograma(j, jornadaActual(j))[i]),
    participantes(cronograma(pre(j), jornadaActual(j))[i]));
asegura ( $\forall c \leftarrow cronograma(j, jornadaActual(j))$ )
  finalizada(c) == True  $\wedge$  incluida(ranking(c), participantes(c))
   $\wedge$  ( $\forall x \leftarrow [0..|ranking(c)| - 2]$ ) capacidadEnOrden(x, c);
asegura asegura ( $\forall c \leftarrow cronograma(j, jornadaActual(j))$ )
  ( $|lesTocoControlAntiDoping(c)| == 1 \wedge lesTocoControlAntiDoping(c)[0] \in atletas(c)$ 
   $\vee (|lesTocoControlAntiDoping(c)| == 0 \wedge |atletas(c)| == 0)$ );
asegura ( $\forall c \leftarrow cronograma(j, jornadaActual(j))$ )
   $|listaDoping(c, atletas(c))| * 20 \leq$ 
   $\sum(|long(lesTocoControlAntiDoping(comp))| comp \leftarrow cronograma(j, jornadaActual(j)))$ );
aux capacidadEnOrden (x:  $\mathbb{Z}$ , c: Competencia) : Bool =
  capacidad(ranking(c)[x], prm(categoria(c)))  $\geq$  capacidad(ranking(c)[x + 1], prm(categoria(c)));
aux listaDoping (c: Competencia, a: [Atleta]) : [Atleta] = [x | x  $\leftarrow$  a, leDioPositivo(c, x)];
}

```

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 ( $l_1, l_2$ : [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 = ( $\forall i, j \leftarrow [0..|l|], i \neq j$ )  $l_i \neq l_j$ ;

```