

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

## 1. Auxiliares definidas por nosotros

Las siguientes funciones auxiliares serán usadas para varios problemas a lo largo del TP. Si la definición de alguna función no aparece en la especificación de un problema, está en esta sección.

```

aux participa (j:JJOO,a:Atleta) : Bool = ( $\exists c \leftarrow \text{competencias}(j)$ )  $a \in \text{participantes}(c)$ ;
aux paises (j:JJOO) : [Pais] = [  $\text{nacionalidad}(a) \mid a \leftarrow \text{atletas}(j)$  ];
aux ganoOroPais (c:Competencia,p:Pais) : Bool =  $|\text{ranking}(c)| \geq 1 \wedge \text{nacionalidad}(\text{ranking}(c)_0) == \text{pais}$ ;
aux ganoPlataPais (c:Competencia,p:Pais) : Bool =  $|\text{ranking}(c)| \geq 2 \wedge \text{nacionalidad}(\text{ranking}(c)_1) == \text{pais}$ ;
aux ganoBroncePais (c:Competencia,p:Pais) : Bool =  $|\text{ranking}(c)| \geq 3 \wedge \text{nacionalidad}(\text{ranking}(c)_2) == \text{pais}$ ;
aux mismaCompetencia (c1,c2:Competencia) : Bool = ( $\text{categoria}(c_1) == \text{categoria}(c_2)$ )
 $\wedge (\text{mismos}(\text{participantes}(c_2), \text{participantes}(c_2))) \wedge (\text{finalizada}(c_1) == \text{finalizada}(c_2))$ 
 $\wedge (\text{finalizada}(c_1) \rightarrow ((\text{ranking}(c_1) == \text{ranking}(c_2)))$ 
 $\wedge (\text{mismos}(\text{lesTocoControlAntiDoping}(c_1), \text{lesTocoControlAntiDoping}(c_2)))$ 
 $\wedge ((\forall a \leftarrow \text{lesTocoControlAntiDoping}(c_1)) \text{leDioPositivo}(c_1, a) == \text{leDioPositivo}(c_2, a))));$ 

```

## 2. Tipos

```

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

```

## 3. 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  $\text{result} \in \text{deportes}(a)$ ;
  asegura  $(\forall d \leftarrow \text{deportes}(a)) \text{capacidad}(a, d) \leq \text{capacidad}(a, \text{res})$ ;
}

```

```

problema entrenarNuevoDeporte (a: Atleta, d: Deporte, c:  $\mathbb{Z}$ ) {
  requiere  $\neg(d \in \text{deportes}(a))$ ;
  requiere  $0 \leq c \leq 100$ ;

  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 nacionalidad(a) == nacionalidad(pre(a));
asegura ciaNumber(a) == ciaNumber(pre(a));
asegura SeAgregoDeporte : mismos(deportes(a), d : deportes(pre(a))) ∧ ordenada(deportes(a));
asegura LosOtrosNoCambiaron : (∀x ← deportes(a), x ≠ d) capacidad(a, x) == capacidad(pre(a), x);
asegura CapacidadCorrecta : capacidad(a, d) == c;
}

```

## 4. 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) ∧ a ∈ lesTocoControlAntiDoping(c);

  invariante participaUnaSolaVez : sinRepetidos(ciaNumbers(participantes(c)));
  invariante participantesPertenecenACat :
    (∀p ← participantes(c)) prm(categoria(c)) ∈ deportes(p) ∧ sgd(categoria(c)) == sexo(p);
  invariante elRankingEsDeParticipantesYNoHayRepetidos :
    finalizada(c) ⇒ incluida(ranking(c), participantes(c));
  invariante seControlanParticipantesYNoHayRepetidos :
    finalizada(c) ⇒ incluida(lesTocoControlAntiDoping(c), participantes(c));
}

problema finalizarCompetencia (c: Competencia, posiciones: [Atleta], control: [(Atleta, Bool)]) {
  requiere ¬finalizada(c);
  requiere PosicionesCumplenInvariante : incluida(posiciones, participantes(c));
  requiere ControladosCumplenInvariante : incluida(atletasControlados(control), participantes(c));

  modifica c;

  asegura (categoria(c) == categoria(pre(c))) ∧ mismos(participantes(c), participantes(pre(c)));
  asegura finalizada(c);
  asegura RankingCorrecto : ranking(c) == posiciones;
  asegura ControlCorrecto : mismos(lesTocoControlAntiDoping(c), atletasControlados(control));
  asegura ResultadoAntiDopingCorrecto : (∀a ← control) sgd(a) == leDioPositivo(c, prm(a));

  aux atletasControlados (c:[(Atleta,Bool)]) : [Atleta] = [prm(a) | a ← c];
}

problema linfordChristie (c: Competencia, a: Atleta) {
  requiere ¬finalizada(c);
  requiere a ∈ participantes(c);

  modifica c;

  asegura ¬finalizada(c) ∧ categoria(c) == categoria(pre(c));
  asegura AtletaDescalificado : mismos(participantes(pre(c)), a : participantes(c));
}

problema gananLosMasCapaces (c: Competencia) = res : Bool {
  requiere finalizada(c);

  asegura res == (∀i ← [1..|ranking(c)| - 1])
    capacidadCompetencia(c, ranking(c)i) ≥ capacidadCompetencia(c, ranking(c)i+1);

  aux capacidadCompetencia (c:Competencia,a:Atleta) : ℤ = capacidad(a, prm(categoria(c)));
}

problema sancionarTramposos (c: Competencia) {
  requiere finalizada(c);

  modifica c;
}

```

```

asegura finalizada(c) ∧ categoria(c) == categoria(pre(c)) ∧ mismos(participantes(c), participantes(pre(c)));
asegura ranking(c) == rankingSinTramposos(pre(c));
asegura mismos(lesTocoControlAntiDoping(c), lesTocoControlAntiDoping(pre(c)));
asegura (∀a ← lesTocoControlAntiDoping(c)) leDioPositivo(c, a) == leDioPositivo(pre(c), a);

aux rankingSinTramposos (c:Competencia) : [Atleta] = [a | a ← ranking(c),
    a ∈ lesTocoControlAntiDoping(c) → leDioPositivo(c, a) == False];
}

```

## 5. 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);
}

problema dePaseo (j: JJOO) = res : [Atleta] {
    asegura mismos(res, noParticipan(j));
    aux noParticipan (j:JJOO) : [Atleta] = [a | a ← atletas(j), ¬participa(j, a)];
}

problema medallero (j: JJOO) = res : [(Pais, [ℤ])] {
    asegura OrdenadoPorOro : (∀i ← [0..|res| - 1]) sgd(resi)0 ≥ sgd(resi+1)0;
    asegura OrdenadoPorPlata : (∀i ← [0..|res| - 1], sgd(resi)0 == sgd(resi+1)0) sgd(resi)1 ≥ sgd(resi+1)1;
    asegura OrdenadoPorBronce : (∀i ← [0..|res| - 1], sgd(resi)1 == sgd(resi+1)1) sgd(resi)2 ≥ sgd(resi+1)2;
    asegura NoHayPaisesRepetidos : sinRepetidos([prm(p) | p ← res]);
    asegura TodosGanaronMedallas : (∀p ← res) ∑ sgd(p) > 0;
    asegura EstanTodosLosQueGanaron : (∀p ← paises(j), totalMedallasPais(j, p) > 0) ((∃q ← res) prm(q) == p);
    asegura CantidadCorrectaDeMedallas : (∀p ← res)
        ((totalOroPais(j, prm(p)) == sgd(p)0) ∧ (totalPlataPais(j, prm(p)) == sgd(p)1)
        ∧ (totalBroncePais(j, prm(p)) == sgd(p)2));

    aux totalOroPais (j:JJOO,p:Pais) : ℤ = |[c | c ← competencias(j), finalizada(c) ∧ ganoOroPais(c, p)]|;
    aux totalPlataPais (j:JJOO,p:Pais) : ℤ = |[c | c ← competencias(j), finalizada(c) ∧ ganoPlataPais(c, p)]|;
    aux totalBroncePais (j:JJOO,p:Pais) : ℤ = |[c | c ← competencias(j), finalizada(c) ∧ ganoBroncePais(c, p)]|;
    aux totalMedallasPais (j:JJOO,p:Pais) : ℤ = totalOroPais(j, p) + totalPlataPais(j, p) + totalBroncePais(j, p);
}

problema boicotPorDisciplina (j: JJOO, cat: (Deporte, Sexo), p: Pais) = res : ℤ {
    requiere CategoriaValida : (∃c ← competencias(j)) categoria(c) == cat;

    modifica j;

    asegura año(j) == año(pre(j)) ∧ cantDias(j) == cantDias(pre(j)) ∧ jornadaActual(j) == jornadaActual(pre(j)) ∧
        mismos(atletas(j), atletas(pre(j)));
    asegura NoModificaOtrasCompetencias : (∀d ← [1..cantDias(j)])
        ((∀c1 ← cronograma(pre(j), d), categoria(c1) ≠ cat) ((∃c2 ← cronograma(j, d)) mismaCompetencia(c1, c2)));
    asegura NoCreaCompetenciasNuevas : (∀d ← [1..cantDias(j)])
        ((∀c1 ← cronograma(j, d), categoria(c1) ≠ cat) ((∃c2 ← cronograma(pre(j), d)) mismaCompetencia(c1, c2)));
    asegura (∀d ← [1..cantDias(j)]) |cronograma(j, d)| == |cronograma(pre(j), d)|;
    asegura CatFueModificada : (∃c1 ← competencias(j), categoria(c1) == cat)
        ((∃c2 ← competencias(pre(j)), categoria(c2) == cat)
        (mismos(participantes(c1), sacarPais(p, participantes(c2))))
        ∧ (finalizada(c1) == finalizada(c2)))
}

```

```

    ∧ (finalizada(c1) → ((ranking(c1) == sacarPais(p, ranking(c2)))
    ∧ (mismos(lesTocoControlAntiDoping(c1), sacarPais(p, lesTocoControlAntiDoping(c2))))
    ∧ ((∀a ← lesTocoControlAntiDoping(c1)) leDioPositivo(c1, a) == leDioPositivo(c2, a)))));
asegura ResultadoCorrecto : res == |[a | a ← atletas(j), nacionalidad(a) == p ∧ participa(j, a)]|;
aux sacarPais (p:Pais, l:[Atleta]) : [Atleta] = [a | a ← l, nacionalidad(a) ≠ p];
}

problema losMasFracasados (j: JJOO, p: Pais) = res : [Atleta] {
  asegura Participaron : (∀a ← res) a ∈ losQueMasParticiparon(j, p);
  asegura NoGanaronNada : (∀a ← res) totalMedallasAtleta(j, a) == 0;
  asegura EstanTodos : (∀a ← losQueMasParticiparon(j, p), totalMedallasAtleta(j, a) == 0) a ∈ res;

  aux losQueMasParticiparon (j:JJOO, p:Pais) : [Atleta] = [a | a ← atletas(j), nacionalidad(a) == p
    ∧ ((∀b ← atletas(j), nacionalidad(b) == p) cantCompetencias(j, a) ≥ cantCompetencias(j, b))];
  aux cantCompetencias (j:JJOO, a:Atleta) : ℤ = |[c | c ← competencias(j), a ∈ participantes(c)]|;
  aux totalMedallasAtleta (j:JJOO, a:Atleta) : ℤ =
    |[c | c ← competencias(j), finalizada(c) ∧ ganoMedallaAtleta(c, a)]|;
  aux ganoMedallaAtleta (c:Competencia, a:Atleta) : Bool = (|ranking(c)| ≥ 1 ∧ ranking(c)0 == a)
    ∨ (|ranking(c)| ≥ 2 ∧ ranking(c)1 == a) ∨ (|ranking(c)| ≥ 3 ∧ ranking(c)2 == a);
}

problema liuSong (j: JJOO, a: Atleta, p: País) {
  requiere a ∈ atletas(j);

  modifica j;

  asegura año(j) == año(pre(j)) ∧ cantDias(j) == cantDias(pre(j)) ∧ jornadaActual(j) == jornadaActual(pre(j));
  asegura SoloCambioElAtletaDeseado : (∀d ← [1..cantDias(j)])
    ((∀c1 ← cronograma(pre(j), d)) ((∃c2 ← cronograma(j, d)) mismaCompNacCamb(c1, c2, a, p))
    ∧ |cronograma(j, d)| == |cronograma(pre(j), d)|);

  aux mismoNacCamb (a1, a2:Atleta, p:Pais) : Bool = nombre(a1) == nombre(a2) ∧ sexo(a1) == sexo(a2)
    ∧ añoNacimiento(a1) == añoNacimiento(a2) ∧ nacionalidad(a2) == p ∧ ciaNumber(a1) == ciaNumber(a2)
    ∧ deportes(a1) == deportes(a2) ∧ ((∀d ← deportes(a1)) capacidad(a1, d) == capacidad(a2, d));
  aux mismosAtlNacCamb (l1, l2: [Atleta], a:Atleta, p:Pais) : Bool =
    ((∀a1 ← l1, ciaNumber(a1) ≠ ciaNumber(a)) ((∃a2 ← l2) a1 == a2))
    ∧ ((∃a1 ← l1, ciaNumber(a1) == ciaNumber(a)) → ((∃a2 ← l2) mismosNacCamb(a1, a2, p)))
    ∧ |l1| == |l2|;
  aux mismaCompNacCamb (c1, c2:Competencia, a:Atleta, p:Pais) : Bool =
    categoria(c1) == categoria(c2) ∧ mismosAtlNacCamb(participantes(c1), participantes(c2), a, p)
    ∧ finalizada(c1) == finalizada(c2)
    ∧ (finalizada(c1) → (mismosAtlNacCamb(ranking(c1), ranking(c2), a, p)
    ∧ mismoOrden(ranking(c1), ranking(c2))
    ∧ mismosAtlNacCamb(lesTocoControlAntiDoping(c1), lesTocoControlAntiDoping(c2), a, p)
    ∧ ((∀b ← lesTocoControlAntiDoping(c1), ciaNumber(b) ≠ ciaNumber(a))
    leDioPositivo(c1, b) == leDioPositivo(c2, b))
    ∧ ((∃a1 ← lesTocoControlAntiDoping(c1), a2 ← lesTocoControlAntiDoping(c2)) mismoNacCamb(a1, a2, p)
    → leDioPositivo(c1, a1) == leDioPositivo(c2, a2)))));
  aux mismoOrden (l:s:[Atleta]) : Bool = (∀i ← [0..|l|]) ciaNumber(li) == ciaNumber(si);
}

problema stevenBradbury (j: JJOO) = res : Atleta {
  asegura res ∈ ganaronOro(j);
  asegura MenosCapaz : (∀a ← ganaronOro(j)) menorCapOro(j, a) ≥ menorCapOro(j, res);

  aux menorCapOro (j:JJOO, a:Atleta) : ℤ = minimo(capacidadOro(j, a));
  aux capacidadOro (j:JJOO, a:Atleta) : [ℤ] = [capacidad(a, d) | d ← deportesOro(j, a)];
  aux deportesOro (j:JJOO, a:Atleta) : [Deporte] =
    [prm(categoria(c)) | c ← competencias(j), finalizada(c) ∧ ganoOroAtleta(c, a)];
  aux ganoOroAtleta (c:Competencia, a:Atleta) : Bool = |ranking(c)| ≥ 1 ∧ ranking(c)0 == a;
}

problema uyOrdenadoAsíHayUnPatrón (j: JJOO) = res : Bool {
  asegura res == hayPatron(mejoresPaises(j));

  aux hayPatron (ps:[Pais]) : Bool = (∀i, j ← [0..|ps|], i ≡ j (|sacarRepetidos(ps)|)) psi == psj;
  aux mejoresPaises (j:JJOO) : [Pais] = [paisDelDia(cronograma(j, d)) | d ← [1..jornadaActual(j)]];
}

```

```

aux paisDelDia (cs:[Competencia]) : Pais =
  cab[p | p ← paísesCompetidores(cs), ((∀q ← paísesCompetidores(cs))totalOroComp(cs,p) ≥ totalOroComp(cs,q)) ∧
aux paísesCompetidores (cs:[Competencia]) : [Pais] = [nacionalidad(a) | a ← participantes(c), c ← (cs)] ;
aux totalOroComp (cs:[Competencia], p:Pais) : ℤ = |[c | c ← cs, finalizada(c) ∧ ganoOroPais(c,p)]| ;
aux sacarRepetidos (l:[T]) : [T] = [li | i ← [0..|l|], li ∉ l[0..i)] ;
}

problema sequiaOlimpica (j: JJOO) = res : [País] {
  asegura SonLosDeMayorSequia : (∀p ← res)((∀q ← países(j))mayorSequia(j,p) ≥ mayorSequia(j,q)) ;
  asegura EstanTodos : (∀p ← países(j), ((∀q ← países(j))mayorSequia(j,p) ≥ mayorSequia(j,q)))p ∈ res ;

  aux mayorSequia (j:JJOO, p:Pais) : ℤ = cab([k - i | i, k ← [1..jornadaActual(j)], i < k ∧ haySequia(j,p,i,k) ∧
    ((∀l, m ← [1..jornadaActual(j)], l < m ∧ haySequia(j,p,l,m))k - i ≥ m - l)] ) ;
  aux haySequia (j:JJOO, p:Pais, i,k:ℤ) : Bool = (∀d ← [i..k])(∀c ← cronograma(j,d))¬ganoMedallaPais(c,p) ;
  aux ganoMedallaPais (c:Competencia, p:Pais) : Bool =
    ganoOroPais(c,p) ∨ ganoPlataPais(c,p) ∨ ganoBroncePais(c,p) ;
  aux : = ;
}

problema transcurrirDia (j: JJOO) {
}

```

## 6. Auxiliares

```

aux ciaNumbers (as: [Atleta]) : [ℤ] = [ciaNumber(a) | a ← as] ;
aux competencias (j: JJOO) : [Competencia] = [c | d ← [1..cantDias(j)], c ← cronograma(j,d)] ;
aux incluida (l1, l2: [T]) : Bool = (∀x ← l1)cuenta(x, l1) ≤ cuenta(x, l2) ;
aux lasPasadasFinalizaron (j: JJOO) : Bool = (∀d ← [1..jornadaActual(j)))(∀c ← cronograma(j,d))finalizada(c) ;
aux lasQueNoPasaronNoFinalizaron (j: JJOO) : Bool =
  (∀d ← (jornadaActual(j)..cantDias(j)))(∀c ← cronograma(j,d))¬finalizada(c) ;
aux ordenada (l:[T]) : Bool = (∀i ← [0..|l| - 1])li ≤ li+1 ;
aux sinRepetidos (l: [T]) : Bool = (∀i, j ← [0..|l|], i ≠ j)li ≠ lj ;

```