

Algoritmos y Estructura de Datos I

Primer cuatrimestre 2012

Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires

TP OJOTA (Organización de Juegos Olímpicos) v1.0

Grupo 4

Integrante	LU	Correo electrónico
Nicolas Lasso	763/10	lasso.nico@gmail.com
Guido Tripodi	843/10	guido.tripodi@hotmail.com
Tomas Agustin Shaurli	671/10	zeratulzero@hotmail.com
Patricio Inzaghi	255/11	pinzaghi@dc.uba.ar

1. Aclaraciones

1. En el ejercicio 7, en caso de empate en medallas de oro, se compara las medallas de plata, si siguen siendo las mismas, se compara las medallas de bronce. En caso de tener las 3 medallas iguales se considera mejor al primero que se comparo. Si no hay ningun pais con medallas la lista resultante sera vacia.
2. En el ejercicio 12, consideramos a un patron, como la secuencia de paises hasta que aparece un elemento repetido sin incluirlo. Por Ej: [Argentina, USA, Canada, Argentina, Chile] el patron seria [Argentina, USA, Canada], en cambio en [Argentina, USA, USA, Argentina, Chile] el patron seria [Argentina, USA].

2. Resolucion

1. problema `entrenarNuevoDeporte` (a: Atleta, d: Deporte, c: \mathbb{Z}) {
 requiere $d \notin \text{deportes}(\text{pre}(a))$;
 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 $\text{naionalidad}(a) == \text{nacionalidad}(\text{pre}(a))$;
 asegura $\text{ciaNumber}(a) == \text{ciaNumber}(\text{pre}(a))$;
 asegura $\text{mismos}(\text{deportes}(a), \text{deportes}(\text{pre}(a)) : d)$;
 asegura $(\forall d \in \text{deportes}(\text{pre}(a))) \text{capacidad}(a, d) == \text{capacidad}(\text{pre}(a), d)$;
 asegura $\text{capacidad}(a, d) == c$;
}
2. problema `finalizarCompetencia` (c: Competencia, posiciones: [Atleta], control: [(Atleta, Bool)]) {
 requiere $|\text{participantes}(c)| > 0$;
 requiere $(\forall a \in \text{control}) \text{prm}(a) \in \text{participantes}(c)$;
 requiere $(\forall a \in \text{posiciones}) a \in \text{participantes}(c)$;
 requiere $\text{finalizada}(c) == \text{False}$;
 modifica c ;
 asegura $\text{categoria}(c) == \text{categoria}(\text{pre}(c))$;
 asegura $\text{mismos}(\text{participantes}(c), \text{participantes}(\text{pre}(c)))$;
 asegura $\text{finalizada}(c)$;
 asegura $\text{ranking}(c) == \text{posiciones}$;
 asegura $\text{mismos}([\text{prm}(d) | d \leftarrow \text{control}], \text{lesTocoControlAntiDoping}(c))$;
 asegura $(\forall x \leftarrow \text{control}), \text{lesDioPositivo}(c, \text{prm}(x)) == \text{sgd}(x)$;
}
3. problema `linfordChristie` (c: Competencia, a: Atleta) {
 requiere $\text{finalizada}(c) == \text{False}$;
 requiere $\text{atletaPertenezca} : a \in \text{participantes}(c)$;
 modifica c ;
 asegura $\text{categoria}(c) == \text{categoria}(\text{pre}(c))$;
 asegura $\text{finalizada}(c) == \text{finalizada}(\text{pre}(c))$;
 asegura $\text{mismos}(a : \text{participantes}(c), \text{participantes}(\text{pre}(c)))$;
}
4. problema `gananLosMasCapaces` (c: Competencia) = result : Bool {
 requiere $\text{finalizada}(c) == \text{true}$;
 asegura $(\forall a \in \text{ranking}(c)) a \in \text{participantes}(c)$;
 asegura $\text{Result} == \text{masCapaces}(c)$;
}

5. problema sancionarTramposos (c: Competencia) {
 requiere *finalizada*(c) == true;
 modifica c;
 asegura *categoría*(c) == *categoría*(pre(c));
 asegura *finalizada*(c) == *finalizada*(pre(c));
 asegura *mismos*(participantes(c), participantes(pre(c)));
 asegura *mismos*(lesTocoControlAntiDoping(c), lesTocoControlAntiDoping(pre(c)));
 asegura ($\forall a \in \text{participantes}(c)$) *leDioPositivo*(c, a) == *leDioPositivo*(pre(c), a);
 asegura *ranking*(c) == *borrarPositivos*(pre(c));
 aux *borrarPositivos* (c:competencia) : [Atleta] = $[x | x \leftarrow \text{ranking}(c), x \notin \text{lesTocoControlAntiDoping}(c) \vee (x \in \text{lesTocoControlAntiDoping}(c) \wedge \text{leDioPositivo}(c, x) == \text{False})]$;
}
6. problema clasificóTarde (c:Competencia, a:Atleta) {
 requiere *finalizada*(c) == True;
 requiere $a \notin \text{participantes}(c)$;
 requiere ($\exists d \leftarrow \text{deportes}(a)$), $d == \text{prm}(\text{categoría}(c)) \wedge \text{sexo}(a) == \text{sgd}(\text{categoría}(c))$;
 modifica c;
 asegura *categoría*(c) == *categoría*(pre(c));
 asegura *mismos*(participantes(c), participantes(pre(c)) : a);
 asegura *finalizada*(c) == *finalizada*(pre(c));
}
7. problema dePaseo (j: JJOO) = result : [Atleta] {
 asegura *mismos*(result, noParticiparon(j));
 aux noParticiparon (j: JJOO) : [Atleta] = $[x | k \leftarrow \text{competencias}(j), x \leftarrow \text{atletas}(j), x \notin \text{participantes}(k)]$;
}
8. problema medallero (j: JJOO) = result : [(País, [Z])] {
 asegura *mismos*(result, extraerSinMedallas(j));
 asegura *soloTresTiposDeMedallas* : ($\forall p \leftarrow \text{result}, |\text{sgd}(p)| == 3$);
 asegura *paísesConMedallas* : ($\forall p \leftarrow \text{result}, \text{algunaMedalla}(\text{sgd}(p))$);
 asegura *mejoresMedallasQueSiguiente* :
 ($\forall x \leftarrow [0..|\text{result}| - 1], \text{mejoresOIgualesMedallas}(\text{sgd}(\text{result}_x), \text{sgd}(\text{result}_{x+1}))$);
 aux mejoresOIgualesMedallas (m: [Z], p: [Z]) : Bool = *superaPorOro*(m, p);
 aux superaPorOro (m: [Z], p: [Z]) : Bool =
 ifThenElse($m_0 > p_0$, True, *ifThenElse*($m_0 == p_0$, *superaPorPlata*(m, p), False));
 aux superaPorPlata (m: [Z], p: [Z]) : Bool =
 ifThenElse($m_1 > p_1$, True, *ifThenElse*($m_1 == p_1$, *superaPorBronce*(m, p), False));
 aux superaPorBronce (m: [Z], p: [Z]) : Bool =
 ifThenElse($m_0 > p_0$, True, $m_0 == p_0$);
 aux algunaMedalla (m: [Z]) : Bool = ($m_0 \neq 0 \vee m_1 \neq 0 \vee m_2 \neq 0$);
}
9. problema boicotPorDisciplina (j: JJOO, cat: (Deporte, Sexo), p: País) = result : Z {
 requiere ($\exists c \in \text{competencias}(j)$) *categoría*(c) == cat;
 modifica j;
 asegura *año*(j) == *año*(pre(j));
 asegura *cantDias*(j) == *cantDias*(pre(j));
 asegura *mismos*(atletas(j), atletas(pre(j));
 asegura ($\forall d \leftarrow [0.. \text{cantDias}(j)]$) *mismos*(cronograma(j, d), cronograma(pre(j), d));
 asegura *jornadaactual*(j) == *jornadaactual*(pre(j));
}

- asegura $result == participantesDeLaCat(j, cat, p)$;
 asegura $(\forall c \leftarrow competencias(j), categoria(c) == cat), (\forall a \leftarrow atletasDelPais(j, p)) a \notin participantes(c)$;
 aux participantesDeLaCat (j:JJOO, cat: (Deporte, Sexo), p: País) : $\mathbb{Z} = long([i|i \leftarrow atletasDelPais(j, p), x \leftarrow competencias(j), categoria(x) == cat \wedge i \in participantes(x)])$;
 }
10. problema losMasFracasados (j: JJOO, p: País) = result : [Atleta] {
 asegura mismos(result, masFracasados(j, p)) ;
 aux competenciasAtleta (j:JJOO, p:País, a:Atleta) : [Competencia] = $[x|x \leftarrow competencias(j), (\exists p \in participantes(x)) p == a \wedge en(prm(categoria(x)), deportes(a))]$;
 aux atletasFracasados (j:JJOO, p:País) : [(atleta, \mathbb{Z})] = $[(y, |competenciasAtleta(j, p, y)|) | y \leftarrow atletasDelPais(j, p), c \leftarrow competenciasAtleta(j, y, p), \neg en(y, sub(ranking(c), 0, 2))]$;
 aux masFracasados (j:JJOO, p:País) : [Atleta] = $[prm(y)|y \leftarrow atletasFracasados(j, p), k \leftarrow atletasFracasados(j, p), sgd(y) \geq sgd(k)]$;
 }
11. problema liuSong (j: JJOO, a: Atleta, p: País) {
 requiere $a \in atletas(j)$;
 modifica j ;
 asegura $año(j) = año(pre(j))$;
 asegura $cantDias(j) == cantDias(pre(j))$;
 asegura $jornadaActual(j) == jornadaActual(pre(j))$;
 asegura $(\forall d \leftarrow [1..cantDias(j)]) mismos(cronograma(j, d), cronograma(pre(j), d))$;
 asegura $(\forall h \leftarrow atletas(j), ciaNumber(h) == ciaNumber(a)) nacionalidad(h) == p$;
 asegura $|atletas(j)| == |atletas(pre(j))|$;
 asegura $(\forall f \leftarrow atletas(j), ciaNumber(f) \neq ciaNumber(a)) (\exists b \leftarrow atletas(pre(j))) f == b$;
 asegura $(\forall c \leftarrow competencias(j), q \leftarrow participantes(c), ciaNumber(q) == ciaNumber(a)) nacionalidad(q) == p$;
 asegura $(\forall c \leftarrow competencias(j), p \leftarrow participantes(c), h \leftarrow participantes(h), p \neq h, ciaNumber(p) \neq ciaNumber(h))$;
 asegura () ;
 asegura $(\forall c \leftarrow competencias(j), q \leftarrow ranking(c), finalizada(c) == True, ciaNumber(q) == ciaNumber(a)) nacionalidad(q) == p$;
 asegura $(\forall c \leftarrow competencias(j), q \leftarrow lesTocoControlAntiDoping(c), finalizada(c) == True, ciaNumber(q) == ciaNumber(a)) nacionalidad(q) == p$;
 asegura mismos(atletas(j), reemplazaAtleta(a, atletas(pre(j)))) ;
 aux reemplazaAtleta (a: Atleta, ls: [Atleta]) : [Atleta] = $[i \text{ if } ciaNumber(x) == ciaNumber(a) \text{ then } a \text{ else } x | x \in ls]$;
 }
12. problema stevenBradbury (j: JJOO) = result : Atleta {
 requiere algunaCompetenciaFinalizada(j) ;
 asegura esElMenosCapaz :
 $(\forall a \in atletasConOro(j), a \neq result) sumaDeCapacidades(result) \leq sumaDeCapacidades(a)$;
 aux atletasConOro (j:JJOO) : [Atleta] = $[ranking(c)_0 | c \in competencias(j), finalizada(c) \wedge |ranking(c)| > 0]$;
 aux sumaDeCapacidades (a:Atleta) : $\mathbb{Z} = \sum [capacidad(a, d) | d \in deportes(a)]$;
 }
13. problema uyOrdenadoAsíHayUnPatrón (j: JJOO) = result : Bool {
 asegura $result == verificaPaises(j)$;
 aux verificaPaises (j:JJOO) : Bool = $\forall i \in [0..|paisesMasGrosos(j)| - 1], paisesMasGrosos(j)_i == paisesMasGrosos(j)_{i+|patronDePaises(j)|} \wedge ((i + |patronDePaises(j)|) \leq |paisesMasGrosos(j)|)$;
 aux patronDePaises (j:JJOO) : [Pais] = $[sub(paisesMasGrosos(j), 0, i - 2) | i \leftarrow [0..|paisesMasGrosos(j)| - 1], \neg sinRepetidos(sub(paisesMasGrosos(j), 0, i))]_0$;
 aux paisesMasGrosos (j:JJOO) : [Pais] = $[p | x \leftarrow mejoresPaises(j), p \leftarrow sgd(x), i \leftarrow [0..|sgd(x)| - 2], p \leq sgd(x)_{i+1}]$;

```

aux mejoresPaises (j:JJO) : [(Z, [Pais])] = [(dia, p)|x ← paisesPorDia(j), dia ← prm(x), p ← sgd(x),
  i ← [0..|sgd(x)| - 2], (cuenta(p, sgd(x)) ≥ cuenta(sgd(x)i+1, sgd(x)))] ;
aux paisesPorDia (j:JJO) : [(Z, [Pais])] = [(dia, p)|x ← oroPorDia(j), dia ← prm(x), h ← sgd(x),
  i ← [0..|h| - 1], p ← nacionalidad(hi)] ;
aux oroPorDia (j:JJO) : [(Z, [Atleta])] = [(dia, a)|h ← competenciasPorDia(j), dia ← prm(h), x ← sgd(h),
  i ← [0..|x| - 1], a ← ranking(xi)0] ;
aux competenciasPorDia (j:JJO) : [(Z, [Competencia])] = [(dia, c)|dia ← [1..jornadaActual(j)],
  c == mismos(cronograma(j, dia)))] ;
}

problema uyOrdenadoAsíHayUnPatrón (j: JJO) = result : Bool {
asegura result == verificaPaises(j) ;
aux medallasDeOro (j:JJO, p:Pais, d:Dia) : Z = long([x|x ← atletasDelPais(j, p), y ← cronograma(j, d), x ∈
  participantes(y) ∧ x == ranking(y)0]) ;
aux paisesDelDia (j:JJO, d:Dia) : [Pais] = [nacionalidad(x)|y ← cronograma(j, d), x ← participantes(y)] ;
aux mejoresPaises (j:JJO) : [Pais] = [p|d ← [1..jornadaActual(j)]p ← extraerRepetidos(PaisesDelDia(j, d)), q ←
  extraerRepetidos(PaisesDelDia(j, d)), medallasDeOro(j, p, d) ≥ medallasDeOro(j, q, d) ∧ p > q] ;
aux patronDePaises (j:JJO) : [Pais] = [sub(mejoresPaises(j), 0, i - 2)|i ← [0..|mejoresPaises| - 1],
  ¬sinRepetidos(sub(mejoresPaises(j), 0, i))]0] ;
aux verificaPaises (j:JJO) : Bool = ∀i ∈ [0..|mejoresPaises(j)| - 1],
  ((i + |patronDePaises(j)|) ≤ |mejoresPaises(j)|) ∧
  paisesMasGrosos(j)i == mejoresPaises(j)i+|patronDePaises(j)|] ;
}

```

```

14. problema sequíaOlímpica (j: JJO) = result : [Pais] {
asegura mismos(result, extraerRepetidos(elPeorPais(j))) ;
aux atletasDelPais (j:JJO, p: Pais) : [Atleta] = [x|x ← atletas(j), nacionalidad(x) == p] ;
aux paisesConMedallas (J:JJO) : [(Pais, [Bool])] = [(nacionalidad(x), [alguna(h ← (ranking(y)[0, 1, 2])
  k ← (atletasDelPais(j, nacionalidad(x))), k == h))]|x ← atletas(j), i ← [0..cantDias(j)], y ← cronograma(j, i)] ;
aux lugaresDistintos (j:JJO, b:[Bool]) : [Z] = [x + 1|x ← [0..|b| - 1], bx ≠ bx+1] ;
aux paisesConDiasSeparados (j:JJO) : [(Pais, [[Bool]])] = [(prm(x), [sub(sgd(x), cons(0, lugresDistintos(j, sgd(x)))h,
  (cons(0, lugresDistintos(j, sgd(x)))h+1))]|x ← paisesConMedallas(j),
  h ← [0..|cons(0, lugresDistintos(j, sgd(x)))| - 2]]] ;
aux elijoLosFracasados (j:JJO) : [(pais, [[Bool]])] = [(prm(x), [y]
  |x ← paisesConDiasSeparados(j), (∀y ← sgd(x), h ← y)(h == False))] ;
aux mayorCantidadDeDias (j:JJO) : [(pais, Z)] = [(prm(x), long(y))
  |x ← elijoLosFracasados(j), y ← sgd(x), h ← sgd(x), long(y) ≥ long(h)] ;
aux elPeorPais (j:JJO) : [pais] = [prm(x)|x ← mayorCantidadDeDias(j),
  y ← mayorCantidadDeDias(j), sgd(x) ≥ sgd(y)] ;
}

```

```

15. problema transcurrirDia (j: JJO) {
requiere jornadaActual(j) < cantDias(j) ;
modifica j ;
asegura cantDias(j) == cantDias(pre(j)) ;
asegura año(j) == año(pre(j)) ;
asegura mismos(atletas(j), atletas(pre(j))) ;
asegura (∀d ∈ [1..cantDias(pre(j))], d ≠ jornadaActual(pre(j))) mismos(cronograma(pre(j), d), cronograma(j, d)) ;
asegura cambiaElDia : jornadaActual(j) == jornadaActual(pre(j)) + 1 ;
asegura rankingSegunMasCapaces : (∀c ∈ cronograma(pre(j), jornadaActual(pre(j))),
  finalizada(c), |participantes(c)| > 0) masCapaces(c) ;
asegura unControlPorCompetencia : (∀c ∈ cronograma(pre(j), jornadaActual(pre(j))), ¬finalizada(c),
  |participantes(c)| > 0)
  |lesTocaControlAntiDoping(c)| == 1 ;
}

```

```

asegura ifThenElse(todosLosAntiDoping > 0, 0 ≤ sonPositivos(j)/todosLosAntiDoping(j) ≤ 0,05, true);
aux SonPositivos (j:JJOO) : ℤ = long([y|x ← cronograma(j, jornadaActual(pre(j))),
    y ← LesTocoControlAntiDoping(x), LeDioPositivo(x, y)]);
aux todosLosAntiDoping (j:JJOO) : ℤ = long(concat([LesTocoControlAntiDoping(x)|
    x ← cronograma(j, jornadaActual(pre(j))), |participantes(x)| > 0]));
}

16. problema deportesNoOlimpicos (j:JJOO) = result : [deporte] {
asegura mismos(result, noCompiten(j, todosLosDeportes(j)));
asegura (∀x ← result, (∃a ← atletas(j))x ∈ deportes(a);
asegura (∀x ← result, y ← competencias(j))x ≠ prm(categoria(y));
aux todosLosDeportes (j:JJOO) : [deporte] = [deportes(a)|a ← atletas(j)];
aux noCompiten (j:JJOO, ds:[deporte]) : [deporte] = [d|d ← extraerRepetidos(ds), c ← competencias(j), d ≠ prm(categoria(c))];
}

17. problema atletaProdigio (js:[JJOO], cat:(Deporte, Sexo)) = result : Atleta {
requiere (∃j ← js, c ← competencias(j))prm(cat) == prm(categoria(c)) ∧ sgd(cat) == sgd(categoria(c));
asegura result ∈ competidoresOroPorCategoria(js, cat);
asegura (∀a ← competidoresOroPorCategoria(js, cat), a ≠ result) añoNacimiento(result) ≥ añoNacimiento(a);
asegura (∃d ∈ deportes(result))d == prm(cat) ∧ sexo(result) == sgd(cat);
aux competidoresOroPorCategoria (js:[JJOO], cat:(Deporte, Sexo)) : [Atleta] = [ranking(c)0|j ← js, c ← competencias(j),
    prm(categoria(c)) == prm(cat) ∧ sgd(categoria(c)) == sgd(cat)];
}

```

3. Tipos

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

4. 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  $\text{sinRepetidos}(\text{deportes}(a))$  ;
  invariante  $\text{ordenada}(\text{deportes}(a))$  ;
  invariante  $\text{capacidadEnRango} : (\forall d \leftarrow \text{deportes}(a)) 0 \leq \text{capacidad}(a, d) \leq 100$  ;
}
```

5. 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  $\text{finalizada}(c)$  ;
  observador lesTocoControlAntiDoping (c: Competencia) : [Atleta] ;
    requiere  $\text{finalizada}(c)$  ;
  observador leDioPositivo (c: Competencia, a: Atleta) : Bool ;
    requiere  $\text{finalizada}(c) \wedge a \in \text{lesTocoControlAntiDoping}(c)$  ;

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

6. 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  $1 \leq \text{dia} \leq \text{cantDias}(j)$  ;
  observador jornadaActual (j: JJOO) :  $\mathbb{Z}$  ;

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

}

7. Auxiliares

```

aux algunaCompetenciaFinalizada (j:JJO) : Bool = ( $\exists x \in competencias(j)$ )  $finalizada(x) == true \wedge |ranking(x)| > 0$ ;
aux ciaNumbers (as: [Atleta]) : [Z] = [ciaNumber(a) | a  $\leftarrow$  as];
aux competencias (j: JJO) : [Competencia] = [c | d  $\leftarrow$  [1..cantDias(j)], c  $\leftarrow$  cronograma(j, d)];
aux competenciasFinalizadas (j: JJO) : [Competencia] = [c | C  $\leftarrow$  [1..cantDias(j)], c  $\leftarrow$  cronograma(j, d), finalizada(c)];
aux incluida (l1, l2: [T]) : Bool = ( $\forall x \leftarrow l_1$ )  $cuenta(x, l_1) \leq cuenta(x, l_2)$ ;
aux lasPasadasFinalizaron (j: JJO) : Bool = ( $\forall d \leftarrow$  [1..jornadaActual(j)]) ( $\forall c \leftarrow$  cronograma(j, d))  $finalizada(c)$ ;
aux lasQueNoPasaronNoFinalizaron (j: JJO) : Bool =
  ( $\forall d \leftarrow$  (jornadaActual(j)..cantDias(j))) ( $\forall c \leftarrow$  cronograma(j, d))  $\neg finalizada(c)$ ;
aux listaPositivos (c: competencia) : [Atleta] = [lesTocoControlAntiDoping(c), lesDioPositivo(c, x) == True];
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|],  $i \neq j$ )  $l_i \neq l_j$ ;
aux masCapaces (c: competencia) : Bool = ( $\forall x \leftarrow$  [0..|ranking(c)| - 1],  $capacidad(ranking(c)_x, prm(categoria(c))) > capacidad(ran$ 
aux atletasDelPais (j: JJO, p: País) : [Atleta] = [x | x  $\leftarrow$  atletas(j), nacionalidad(x) == p];
aux medallasDelPais (j: JJO, p: País, pos: Z) : Z = long([x | x  $\leftarrow$  atletasDelPais(j, p), y  $\leftarrow$  competencias(j),
  x == ranking(y)pos]);
aux todosLosPaises (j: JJO) : [País] = [nacionalidad(x) | x  $\leftarrow$  atletas(j)];
aux extraerRepetidos (ls: [T]) : [T] = [x | x  $\leftarrow$  [0..|ls|],  $\neg en(ls_x, sub(ls, 0, x))$ ];
aux obtenerMedallero (j: JJO) : [(País, [Z])] = [(x, [medallasDelPais(j, x, y)]) | x  $\leftarrow$  extraerRepetidos(todosLosPaises(j)),
  y  $\leftarrow$  [0, 1, 2]];
aux extraerSinMedallas (j: JJO) : [(País, [Z])] = [x | x  $\leftarrow$  obtenerMedallero(j), ( $\exists x \leftarrow$  sgd(x))  $x > 0$ ];
aux atletasdelpaisout (cat: (Deporte, Sexo), p: País, j: JJO) : Bool = (x  $\leftarrow$  [0..|atletas(j)| - 1],  $p(cat(atletas(j)_x)) \neq$ 
  atletas(j)x);

```