UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación
Algoritmos y Estructura de Datos I
Segundo cuatrimestre de 2016
26 de septiembre de 2016

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` $d \in deportes(a)$;
`invariante` $|deportes(a)| > 0$;
`invariante` $sinRepetidos(deportes(a))$;
`invariante` $ordenada(deportes(a))$;
`invariante capacidadEnRango` : $(\forall d \leftarrow deportes(a))0 \leq capacidad(a, d) \leq 100$;
}

`problema especialidad` (this: Atleta) = `res` : Deporte {
`asegura` $res \in deportes(this)$;
`asegura` $(\forall d \leftarrow deportes(this))capacidad(this, res) \geq capacidad(this, d)$;
}

`problema Atleta` (this: Atleta, nom : String, s : Sexo, a : $\mathbb{Z}$, nac : Pais, cia : $\mathbb{Z}$) {
`modifica` $this$;
`asegura` $nombre(this) == nom$;
`asegura` $sexo(this) == s$;
`asegura` $añoNacimiento(this) == a$;
`asegura` $nacionalidad(this) == nac$;
`asegura` $ciaNumber(this) == cia$;
`asegura` $deportes(this) == [\text{“}Tenis\text{”}]$;
`asegura` $capacidad(this, \text{“}Tenis\text{”}) == 50$;
}

`problema nombre` (this : Atleta) = `res` : String {
`asegura` $nombre(this) == result$;
}

`problema sexo` (this : Atleta) = `res` : Sexo {
`asegura` $sexo(this) == result$;
}

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

`problema nacionalidad` (this : Atleta) = `res` : Pais {
`asegura` $nacionalidad(this) == result$;

}

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 $d \in deportes(this)$;
    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 $(\forall x \leftarrow deportes(this), x \neq d) capacidad(this, x) == capacidad(pre(this), x)$;
    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 =
        $deportes(a1) == deportes(a2) \wedge (\forall d \leftarrow deportes(a1)) capacidad(a1, d) == capacidad(a2, d)$;
}

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 $finalizada(c) \wedge a \in lesTocoControlAntiDoping(c)$;

    invariante participaUnaSolaVez : $sinRepetidos(ciaNumbers(participantes(c)))$;
    invariante participantesPertenecenACat :
        $(\forall p \leftarrow participantes(c)) prm(categoria(c)) \in deportes(p) \wedge sgd(categoria(c)) == genero(p)$;
    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 : $(\forall a \leftarrow as) d \in deportes(a) \wedge s == sexo(a)$;
    requiere noHayAtletasRepetidos : $sinRepetidos(ciaNumbers(as))$;
    modifica $this$;
    asegura $categoria(this) == (d, s)$;

```
        asegura mismos(participantes(this), as);
        asegura ¬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 :
            (∀a ← lesTocoControlAntiDoping(this))leDioPositivo(this, a) ⇔ leDioPositivo(pre(this), a) ;

        aux noLoDescubrenDopado (a: Atleta, c: Competencia) : Bool =
            a ∉ lesTocoControlAntiDoping(c) ∨ ¬leDioPositivo(c, a) ;
}

problema operator== (this,c: Competencia) = res : Bool {
        asegura mismosParticipantesYCategoria(this, c) ∧ finalizada(this) == finalizada(c) ∧
            (finalizada(this) ⟶ ranking(this) == ranking(c) ∧ coincidenControlesAntidoping(this, c)) ;

        aux mismosParticipantesYCategoria (c1, c2: Competencia) : Bool = mismos(participantes(c1), participantes(c2))∧
            categoria(c1) == categoria(c2) ;
        aux coincidenControlesAntidoping (c1,c2: Competencia) : Bool =
            mismos(lesTocoControlAntidoping(c1), lesTocoControlAntidoping(c2)) ∧
            (∀a ← lesTocoControlAntidoping(c1))leDioPositivo(c1, a) == leDioPositivo(c2, a) ;
}
```

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) ;
}

problema JJOO (this: JJOO, año: ℤ, as: [Atleta], cron: [[Competencia]]) {
        requiere sinRepetidos(ciaNumbers(as)) ;
        requiere (∀cs ← concat(cron))(¬(∃i, j ← [0..|concat(cron)|), i ≠ j)categoria(cs_i) == categoria(cs_j))) ;
        requiere (∀cs ← concat(cron))incluida(participantes(cs), as) ;
        requiere |cron| ≥ 1 ;
        requiere (∀c ← concat(cron))¬finalizada(c) ;
        modifica this ;
        asegura año == año(this) ;
        asegura mismos(atletas(this), as) ;
        asegura |cron| == cantDias(this) ;
        asegura (∀j ← [0..|cron|))mismos(cron_j, cronograma(this, j + 1)) ;
        asegura jornadaActual(this) == 1 ;
}

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

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

problema cantDias (this: JJOO) = res : ℤ {
        asegura cantDias(this) == result ;
}
```

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

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

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

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

problema dePaseo (this: JJOO) = res : [Atleta] {
    asegura noParticipanEnNinguna : $mismos(res, fueronAPasear(this))$;

    aux fueronAPasear (j: JJOO) : [Atleta] = $[\, a \mid a \leftarrow atletas(j), \neg(\exists c \leftarrow competencias(j)) a \in participantes(c) \,]$;
}

problema medallero (this: JJOO) = res : [(Pais, [$\mathbb{Z}$])] {
    asegura paisesConMedallas : $mismos(primeros(res), paisesQueGanaron(this))$;
    asegura cantidadMedallasCorrecta : $(\forall m \leftarrow res)\, |sgd(m)| == 3 \wedge$
        $sgd(m)_0 == |filtrarPorPais(medallistasOro(this), prm(m))| \wedge$
        $sgd(m)_1 == |filtrarPorPais(medallistasPlata(this), prm(m))| \wedge$
        $sgd(m)_2 == |filtrarPorPais(medallistasBronce(this), prm(m))|$;
    asegura bienOrdenada : $(\forall i \leftarrow (0..|res|))masMedallas(sgd(res_{i-1}), sgd(res_i))$;

    aux paisesQueGanaron (j: JJOO) : [Pais] = $sacarRepetidos(nacionalidades(medallistasOro(j) ++$
        $medallistasPlata(j) ++ medallistasBronce(j)))$;
    aux masMedallas (x, y: [$\mathbb{Z}$]) : 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: JJOO, cat: (Deporte, Sexo), p: Pais) = res : $\mathbb{Z}$ {
    requiere esCategoriaValida : $(\exists c \leftarrow competencias(this))categoria(c) == cat$;
    modifica $this$;
    asegura soloCambiaCronograma : $año(this) == año(pre(this)) \wedge cantDias(this) == cantDias(pre(this))$
        $\wedge\, jornadaActual(this) == jornadaActual(pre(this)) \wedge mismos(atletas(this), atletas(pre(this)))$;
    asegura mismaCantDeCompetencias : $(\forall d \leftarrow [1..cantDias(this)])\, |cronograma(this, d)| == |cronograma(pre(this), d)|$;
    asegura lasOtrasCompetenciasNoCambian : $(\forall d \leftarrow [1..cantDias(this)])(\forall c \leftarrow cronograma(pre(this), d), categoria(c) \neq$
        $cat) laCompetenciaSeMantiene(this, d, c)$;
    asegura boicotAEsaCat : $(\exists c \leftarrow cronograma(this, elDiaDeEsaCat(pre(this), cat)))$
        $igualSalvoBoicot(c, competenciaDeCat(pre(this), cat), p)$;
    asegura $result == |filtrarPorPais(participantes(competenciaDeCat(pre(this), cat)), p)|$;

    aux elDiaDeEsaCat (j: JJOO, cat: (Deporte, Sexo)) : $\mathbb{Z}$ =
        $[\, d \mid d \leftarrow [1..cantDias(j)], (\exists c \leftarrow cronograma(j, d))categoria(c) == cat \,]_0$;
    aux competenciaDeCat (j: JJOO, cat: (Deporte, Sexo)) : Competencia = $[\, c \mid c \leftarrow competencias(j), categoria(c) ==$
        $cat \,]_0$;
    aux igualSalvoBoicot (c, prec: Competencia, p: Pais) : Bool = $categoria(c) == categoria(prec) \wedge$
        $mismos(participantes(c), sacarLosDePais(participantes(prec), p) \wedge finalizada(c) \Leftrightarrow finalizada(prec)$
        $\wedge\, finalizada(c) \Rightarrow (ranking(c) == sacarLosDePais(ranking(prec), p) \wedge$
        $mismos(lesTocoControlAntiDoping(c), sacarLosDePais(lesTocoControlAntiDoping(prec), p))$
        $\wedge\, (\forall a \leftarrow lesTocoControlAntiDoping(c)) leDioPositivo(c, a) \Leftrightarrow leDioPositivo(prec, a))$;
    aux sacarLosDePais (as: [Atleta], p: Pais) : [Atleta] = $[\, a \mid a \leftarrow as, nacionalidad(a) \neq p \,]$;
}

problema losMasFracasados (this: JJOO, p: Pais) = res : [Atleta] {
    asegura $mismos(res, noGanaronMedallas(this, losMasParticipantes(this, atletasDelPais(this, p))))$;

    aux atletasDelPais (j: JJOO, p: Pais) : [Atleta] = $[\, a \mid a \leftarrow atletas(j), nacionalidad(a) == p \,]$;
    aux losMasParticipantes (j: JJOO, as: [Atleta]) : [Atleta] = $[\, a \mid a \leftarrow as,$
        $(\forall x \leftarrow as) cantCompetencias(j, a) \geq cantCompetencias(j, x) \,]$;
    aux cantCompetencias (j: JJOO, a: Atleta) : $\mathbb{Z}$ = $|[\, c \mid c \leftarrow competencias(j), a \in participantes(c) \,]|$;

aux noGanaronMedallas (j: JJOO, as: [Atleta]) : [Atleta] $= [\, a \mid a \leftarrow as, cantMedallas(j, a) == 0 \,]$;
aux cantMedallas (j: JJOO, a: Atleta) : $\mathbb{Z} = |[\, c \mid c \leftarrow competencias(j), estaEnElPodio(c, a) \,]|$;
aux estaEnElPodio (c: Competencia, a: Atleta) : Bool $= finalizada(c) \wedge (salioPrimero(c, a) \vee salioSegundo(c, a) \vee salioTercero(c, a))$;
aux salioPrimero (c: Competencia, a: Atleta) : Bool $= |ranking(c)| \geq 1 \wedge ranking(c)_0 == a$;
aux salioSegundo (c: Competencia, a: Atleta) : Bool $= |ranking(c)| \geq 2 \wedge ranking(c)_1 == a$;
aux salioTercero (c: Competencia, a: Atleta) : Bool $= |ranking(c)| \geq 3 \wedge ranking(c)_2 == a$;
}

problema liuSong (this: JJOO, a: Atleta, p: País) {
requiere estaLiu : $a \in atletas(this)$;
modifica $this$;
asegura loDemasIgual : $año(this) == año(pre(this)) \wedge cantDias(this) == cantDias(pre(this)) \wedge jornadaActual(this) == jornadaActual(pre(this))$;
asegura mismaCantidadAtletas : $|atletas(this)| == |atletas(pre(this))|$;
asegura atletasIguales : $(\forall at1 \leftarrow atletas(pre(this)), \neg(at1 == a))at1 \in atletas(this)$;
asegura cambioLiu : $(\forall at1 \leftarrow atletas(pre(this)), at1 == a)(\exists at2 \leftarrow atletas(this))igualSalvoPais(at1, at2, p)$;
asegura mismaCantDeCompetencias : $(\forall d \leftarrow [1..cantDias(this)])\, |cronograma(pre(this), d)| == |cronograma(this, d)|$;
asegura lasOtrasCompetenciasNoCambian : $(\forall d \leftarrow [1..cantDias(this)])(\forall c \leftarrow cronograma(pre(this), d), a \notin participantes laCompetenciaSeMantiene(this, d, c)$;
asegura cambianLasDeLiu : $(\forall d \leftarrow [1..cantDias(this)])(\forall c \leftarrow cronograma(pre(this), d), a \in participantes(c))$ $(\exists c2 \leftarrow cronograma(this, d))igualSalvoLiu(c, c2, a, p)$;

aux igualSalvoPais (at1: Atleta, at2: Atleta, p: Pais) : Bool $= nombre(at1) == nombre(at2) \wedge$ $sexo(at1) == sexo(at2) \wedge añoNacimiento(at1) == añoNacimiento(at2)$ $\wedge\, ciaNumber(at1) == ciaNumber(at2) \wedge deportes(at1) == deportes(at2)$ $\wedge\, (\forall d \leftarrow deportes(at1))capacidad(at1, d) == capacidad(at2, d) \wedge nacionalidad(at2) == p$;
aux igualSalvoLiu (c1: Competencia, c2: Competencia, a: Atleta, p: Pais) : Bool $= categoria(c1) == categoria(c2)$ $\wedge\, participantesYLiu(c1, c2, a, p) \wedge finalizada(c1) \Leftrightarrow finalizada(c2) \wedge finalizada(c1) \Rightarrow$ $(rankingYLiu(c1, c2, a, p) \wedge mismosControladosYLiu(c1, c2, a, p))$;
aux participantesYLiu (c1: Competencia, c2: Competencia, a: Atleta, p: Pais) : Bool $=$ $|participantes(c1)| == |participantes(c2)|$ $\wedge\, (\forall at1 \leftarrow participantes(c1), at1! = a)at1 \in participantes(c2)$ $\wedge\, (\forall at1 \leftarrow participantes(c1), at1 == a)(\exists at2 \leftarrow participantes(c2))igualSalvoPais(at1, at2, p)$;
aux rankingYLiu (c1: Competencia, c2: Competencia, a: Atleta, p: Pais) : Bool $= |ranking(c1)| == |ranking(c2)|$ $\wedge\, (\forall i \leftarrow [0..|ranking(c1)|), ranking(c1)_i! = a)ranking(c2)_i == ranking(c1)_i$ $\wedge\, (\forall i \leftarrow [0..|ranking(c1)|), ranking(c1)_i == a)igualSalvoPais(ranking(c1)_i, ranking(c2)_i, p)$;
aux mismosControladosYLiu (c1: Competencia, c2: Competencia, a: Atleta, p: Pais) : Bool $=$ $|lesTocoControlAntiDoping(c1)| == |lesTocoControlAntiDoping(c2)| \wedge$ $(\forall at1 \leftarrow lesTocoControlAntiDoping(c1), at1! = a)at1 \in lesTocoControlAntiDoping(c2) \wedge leDioPositivo(c1, at1) == leDioPositivo(c2, at1) \wedge$ $(\forall at1 \leftarrow lesTocoControlAntiDoping(c1), at1 == a)(\exists at2 \leftarrow lesTocoControlAntiDoping(c2))$ $igualSalvoPais(at1, at2, p) \wedge leDioPositivo(c1, at1) == leDioPositivo(c2, at2)$;
}

problema stevenBradbury (this: JJOO) = res : Atleta {
requiere alguienGanoMedalla : $(\exists d \leftarrow [1..jornadaActual(this)])(\exists c \leftarrow cronograma(this, d), finalizada(c))\, |ranking(c)| > 0$;
asegura ganoMedallaDeOro : $res \in primeros(ganadoresPorCategoria(this))$;
asegura elMenosCapaz : $(\forall a \leftarrow primeros(ganadoresPorCategoria(this)))$ $peorDesempeño(res, this) \leq peorDesempeño(a, this)$;

aux ganadoresPorCategoria (j: JJOO) : [(Atleta, (Deporte, Sexo))] $= [\, (ranking(c)_0, categoria(c)) \mid$ $d \leftarrow [1..jornadaActual(j)], c \leftarrow cronograma(j, d), finalizada(c) \wedge |ranking(c)| > 0 \,]$;
aux peorDesempeño (a: Atleta, j: JJOO) : $\mathbb{Z} =$ $minimo([\, capacidad(a, prm(sgd(g))) \mid g \leftarrow ganadoresPorCategoria(j), prm(g) == a \,])$;
}

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

aux losMejoresPaises (j: JJOO) : [Pais] $= [\, mejorEseDia(j, i) \mid i \leftarrow [1..jornadaActual(j)], alguienGanoOro(j, i)) \,]$;
aux mejorEseDia (j: JJOO, d: $\mathbb{Z}$) : Pais $= [\, p \mid p \leftarrow paises(j),$ $\neg(\exists p2 \leftarrow paises(j))(cantOro(j, p2, d) > cantOro(j, p, d) \vee (cantOro(j, p2, d) == cantOro(j, p, d) \wedge p2 < p)) \,]_0$;
aux cantOro (j: JJOO, p: Pais, d: $\mathbb{Z}$) : $\mathbb{Z} = |[\, 1 \mid c \leftarrow cronograma(j, d), finalizada(c)$ $\wedge\, ranking(c) \geq 1 \wedge nacionalidad(ranking(c)_0) == p \,]|$;

aux alguienGanoOro (j: JJOO, d: $\mathbb{Z}$) : Bool $= (\exists c \leftarrow cronograma(j,d))finalizada(c) \land ranking(c) \geq 1$;
aux siguenSiempreElMismoOrden (ps:[Pais]) : Bool $= (\forall i,j \leftarrow [0..|ps|-1), i<j \land ps_i == ps_j)ps_{i+1} == ps_{j+1} \land (\forall i,j \leftarrow [1..|ps|), i<j \land ps_i == ps_j)ps_{i-1} == ps_{j-1}$;
}

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

aux secosOlimpicos (j: JJOO) : [País] $= [\, p \mid p \leftarrow paises(j), masDiasSinMedallas(j,p) == maxDiasSinMedallas(j) \,]$;
aux masDiasSinMedallas (j: JJOO, p: País) : $\mathbb{Z} = maxDif(0 : [\, i \mid i \leftarrow [1..jornadaActual(j)), GanoMedallaEseDia(j,p,i) \,] + +[jornadaActual(j)])$;
aux maxDif (ls:[$\mathbb{Z}$]) : $\mathbb{Z} = max([\, ls_i - ls_{i-1} \mid i \leftarrow [1..|ls|) \,])$;
aux GanoMedallaEseDia (j: JJOO, p: Pais, i: $\mathbb{Z}$) : Bool $= (\exists c \leftarrow cronograma(j,i))$
$(|ranking(c)| \geq 1 \land nacionalidad(ranking(c)_0) == p)$
$\lor (|ranking(c)| \geq 2 \land nacionalidad(ranking(c)_1) == p)$
$\lor (|ranking(c)| \geq 3 \land nacionalidad(ranking(c)_2) == p)$;
aux maxDiasSinMedallas (j: JJOO) : $\mathbb{Z} = max([masDiasSinMedallas(j,p) \mid p \leftarrow paises(j) \,])$;
}

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 lasFinalizadasSeMantinen : $(\forall c \leftarrow cronograma(pre(this), jornadaActual(pre(this))), finalizada(c))$
$laCompetenciaSeMantiene(this, jornadaActual(pre(this)), c)$;
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 $= (\exists x \leftarrow cronograma(j,dia))categoria(x) == categoria(c) \land$
$mismos(participantes(x), participantes(c)) \land finalizada(x) \land mismos(ranking(x), participantes(x)) \land$
$ordenada(reverso(capacidades(ranking(x), deporte(x)))) \land$
$|ranking(x)| \geq 1 \Rightarrow |lesTocoControlAntiDoping(x)| == 1$;
}

problema operator== (this, j: JJOO) = res : Bool {
asegura $result == (año(this) == año(j) \land cantDias(this) == cantDias(j)$
$\land jornadaActual(this) == jornadaActual(j) \land mismos(atletas(this), atletas(j)) \land mismoCronograma(this,j))$;

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]) : [$\mathbb{Z}$] $= [\, ciaNumber(a) \mid a \leftarrow as \,]$;
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 $= (\forall i,j \leftarrow [0..|l|), i \neq j)l_i \neq l_j$;
aux capacidades (as: [Atleta], d: Deporte) : [$\mathbb{Z}$] $= [\, capacidad(a,d) \mid a \leftarrow as \,]$;
aux deporte (c: Competencia) : Deporte $= prm(categoria(c))$;
aux laCompetenciaSeMantiene (j: JJOO, d: $\mathbb{Z}$, c: Competencia) : Bool $=$
$(\exists x \leftarrow cronograma(j,d))categoria(x) == categoria(c) \land mismos(participantes(x), participantes(c))$
$\land finalizada(x) \Leftrightarrow finalizada(c) \land finalizada(x) \Rightarrow (ranking(x) == ranking(c) \land mismosControlados(x,c))$;
aux medallistasOro (j: JJOO) : [Atleta] $= [\, ranking(c)_0 \mid d \leftarrow [1..jornadaActual(j)], c \leftarrow cronograma(j,d),$
$finalizada(c) \land |ranking(c)| \geq 1 \,]$;

`aux medallistasPlata` (j: JJOO) : [Atleta] $= [\, ranking(c)_1 \mid d \leftarrow [1..jornadaActual(j)], c \leftarrow cronograma(j,d), finalizada(c) \wedge |ranking(c)| \geq 2 \,]$;

`aux medallistasBronce` (j: JJOO) : [Atleta] $= [\, ranking(c)_2 \mid d \leftarrow [1..jornadaActual(j)], c \leftarrow cronograma(j,d), finalizada(c) \wedge |ranking(c)| \geq 3 \,]$;

`aux minimo` (l: $[\mathbb{Z}]$) : $\mathbb{Z} = [\, x \mid x \leftarrow l, (\forall y \leftarrow l) x \leq y \,]_0$;

`aux mismosControlados` (c_1, c_2: Competencia) : Bool $=$
$mismos(lesTocoControlAntiDoping(c_1), lesTocoControlAntiDoping(c_2)) \wedge$
$(\forall p \leftarrow lesTocoControlAntiDoping(c_1)) leDioPositivo(c_1, p) \Leftrightarrow leDioPositivo(c_2, p)$;

`aux nacionalidades` (as: [Atleta]) : [Pais] $= [\, nacionalidad(a) \mid a \leftarrow as \,]$;

`aux primeros` (l: [(T,S)]) : [T] $= [\, prm(x) \mid x \leftarrow l \,]$;

`aux reverso` (l: [T]) : [T] $= [\, l_{|x|-i-1} \mid i \leftarrow [0..|l|) \,]$;

`aux sacarRepetidos` (l: [T]) : [T] $= [\, l_i \mid i \leftarrow [0..|l|), l_i \notin l_{[0..i)} \,]$;