DEPARTAMENTO DE COMPUTACION

Facultad de Ciencias Exactas y Naturales - UBA

Trabajo práctico 1

Especificación de TADs

October 27, 2025 — Algoritmos y Estructuras de Datos

Los Simultaneos

Integrante	LU	Correo electrónico
Perez, Jimena Huilen	215/20	jimehperez@gmail.com
Ferigolo, Ayelén Denise	872/22	ayelenferigolo@gmail.com
Elizabe, Elias Gaspar	188/22	eliaselizabe3@gmail.com
Garcia Arauco, Daniela Belen	840/17	belengarciaarauco@gmail.com

Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Ciudad Universitaria - (Pabellón I/Planta Baja)
Intendente Güiraldes 2610 - C1428EGA
Ciudad Autónoma de Buenos Aires - Rep. Argentina
Tel/Fax: (++54 +11) 4576-3300
http://www.exactas.uba.ar

Posicion $ES\ tupla\ <\mathbb{Z},\mathbb{Z}>$

Examen $ES\ seq\ <\mathbb{Z}>$

Estudiante $ES\ struct\ <posicion : Posicion, examen : Examen, entregado : Bool>$

Notas $ES\ seq\ <tupla\ <estudiante : Posicion, notaFinal : \mathbb{R}>>$

TAD EdR {

obs dimensionAula : $\mathbb{Z}$

obs solucionExamen : $Examen$

obs estudiantes : conj$\langle Estudiante \rangle$

proc EdR(in $dimensionAula : \mathbb{Z}$, in $solucionExamen : Examen$, in $cantidadDeEstudiantes : \mathbb{Z}$) : EdR {

requiere {

$solucionValida(solucionExamen) \land$

$cantidadDeEstudiantes > 0 \land$

$dimensionAula > 0 \land$

$aulaValida(dimensionAula, cantidadDeEstudiantes)$

}

asegura {

$(\forall e : Estudiante)\ (e \in res.estudiante \rightarrow$

$esPosicionValida(dimensionAula, e.posicion, res.estudiantes) \land$

$esExamenVacio(e.examen, solucionExamen) \land e.entregado = false) \land$

$res.solucionExamen = solucionExamen \land$

$res.dimensionAula = dimensionAula \land$

$|res.estudiantes| = cantidadDeEstudiantes$

}

}

proc igualdad(in $edr1 : EdR$, in $edr2 : EdR$) : bool {

requiere { $True$ }

asegura {

$res = true \leftrightarrow$

$edr1.dimensionAula = edr2.dimensionAula \land$

$edr1.solucionExamen = edr2.solucionExamen \land$

$mismosEstudiantes(edr1.estudiantes, edr2.estudiantes)$

}

}

proc copiarse(inout $edr : EdR$, in $e_1 : Estudiante$) : {

requiere {

$edr = edr_0 \land$

$e_1 \in edr_0.estudiantes \land$

$e_1.entregado = false \land$

$(\exists e_2 : Estudiante)\ (e_2 \in edr_0.estudiantes \land esEstudianteCopiable(e_1, e_2))$

}

```
    asegura {
        (∃e'_1, e_2 : Estudiante) (e_2 ∈ edr_0.estudiantes ∧ e'_1 ∈ edr.estudiantes ∧
        esEstudianteCopiable(e_1, e_2) ∧
        meCopio(e_1.examen, e'_1.examen, e_2.examen)∧resolvioUnEjercicio(e_1.examen, e'_1.examen)∧

        noSeModificaPosicionYEntregado(e_1, e'_1) ∧
        (∀e : Estudiante) (e ∈ edr_0.estudiantes ∧ e ≠ e_1 → e ∈ edr.estudiantes)) ∧

        edr.solucionExamen = edr_0.solucionExamen ∧
        edr.dimensionAula = edr_0.dimensionAula
    }
}

proc consultarDarkWeb(inout edr : EdR, in N : ℤ, in solucionDarkWeb : Examen) : {
    requiere {
        edr = edr_0 ∧
        esSolucionDeExamen(solucionDarkWeb, edr_0.solucionExamen) ∧
        0 ≤ N
    }
    asegura {
        |edr.estudiantes| = |edr_0.estudiantes| ∧
        (∀e : Estudiante) (e ∈ edr_0.estudiantes ∧ e.entregado = true → e ∈ edr.estudiantes) ∧
        (∀e : Estudiante) (e ∈ edr_0.estudiantes ∧ e.entregado = false → (∃e' : Estudiante) (
        e' ∈ edr.estudiantes ∧ e'.posicion = e.posicion ∧ e'.entregado = e.entregado ∧
        (e'.examen = e.examen ∨ e'.examen = solucionDarkWeb))) ∧

        hayComoMuchoNexamenesDeLaDarkWeb(solucionDarkWeb, N, edr.estudiantes) ∧
        edr.solucionExamen = edr_0.solucionExamen ∧
        edr.dimensionAula = edr_0.dimensionAula
    }
}

proc resolver(inout edr : EdR, in e_1 : Estudiante, in pasos : seq⟨Examen⟩) : seq⟨Examen⟩ {
    requiere {
        edr = edr_0 ∧
        e_1 ∈ edr_0.estudiantes ∧
        noTerminoExamen(e_1) ∧
        pasosValidos(pasos, e_1.examen)
    }
    asegura {
        (∃e'_1 : Estudiante) (e'_1 ∈ edr.estudiantes ∧
        resolvioUnEjercicio(e_1.examen, e'_1.examen) ∧
        noSeModificaPosicionYEntregado(e_1, e'_1) ∧
        res = pasos ++ e'_1.examen  ∧
        (∀e : Estudiante) (e ≠ e_1 ∧ e ∈ edr_0.estudiantes → e ∈ edr.estudiantes))  ∧

        edr.solucionExamen = edr_0.solucionExamen ∧
        edr.dimensionAula = edr_0.dimensionAula ∧
        |res| = |pasos| + 1
    }
}
```

```
proc entregar(inout edr : EdR, in e_1 : Estudiante) : {
    requiere {
        edr = edr_0 ∧
        e_1 ∈ edr_0.estudiantes ∧
        e_1.entregado = False
    }
    asegura {
        (∃e'_1 : Estudiante) (e'_1 ∈ edr.estudiantes ∧ e'_1.entregado = true ∧
        noSeModificaExamenYPosicion(e_1, e'_1)) ∧
        (∀e : Estudiante) (e ∈ edr_0.estudiantes ∧ e ≠ e_1 → e ∈ edr.estudiantes) ∧

        |edr.estudiantes| = |edr_0.estudiantes| ∧
        edr.dimensionAula = edr_0.dimensionAula ∧
        edr.solucionExamen = edr_0.solucionExamen
    }
}

proc chequearCopias(in edr : EdR) : seq⟨Estudiante⟩ {
    requiere { todosEntregaron(edr.estudiantes) }
    asegura {
        (∀e : Estudiante) (e ∈ edr.estudiantes ∧ soySospechoso(e, edr.estudiantes) → e ∈ res) ∧
        ¬(∃r : Estudiante) (r ∈ res∧(¬soySospechoso(r, edr.estudiantes)∨(r ∉ edr.estudiantes)))∧
        ¬hayEstudiantesRepetidos(res)
    }
}

proc corregir(in edr : EdR) : Notas {
    requiere { todosEntregaron(edr.estudiantes) }
    asegura {
        (∀e : Estudiante) (e ∈ edr.estudiantes ∧ ¬soySospechoso(e, edr.estudiantes) →
        < e.posicion, nota(e.examen, edr.solucionExamen) >∈ res) ∧
        sonNotasValidas(res, edr.estudiantes) ∧
        |res| = |edr.estudiantes| − cantidadDeSospechosos(edr.estudiantes)
    }
}

pred solucionValida(examen : Examen) {
    |examen| > 0 ∧ (∀i : ℤ) (0 ≤ i < |examen| →_L 0 ≤ examen[i] ≤ 9)
}

pred aulaValida(dimensionAula : ℤ, cantidadDeEstudiantes : ℤ) {
    cantidadDeEstudiantes ≤ dimensionAula ∗ ⌊(dimensionAula + 1)/2⌋
}

pred esPosicionValida(dimensionAula : ℤ, posicion : Posicion, estudiantes : conj⟨Estudiante⟩)
{
    0 ≤ posicion_0, posicion_1 ≤ dimensionAula ∧
    ¬existeEstudiante(posicion_0, posicion_1, estudiantes) ∧
    ¬existeEstudiante(posicion_0, posicion_1 + 1, estudiantes) ∧
    ¬existeEstudiante(posicion_0, posicion_1 − 1, estudiantes)
}
```

pred existeEstudiante($fila : \mathbb{Z}, columna : \mathbb{Z}, estudiantes : \mathsf{conj}\langle Estudiante\rangle$) {
$(\exists e : Estudiante)\ (e \in estudiantes \wedge < fila, columna >= e.posicion)$
}

pred esExamenVacio($examen : Examen, solucionExamen : Examen$) {
$|examen| = |solucionExamen| \wedge$
$(\forall i : \mathbb{Z})\ (0 \leq i < |examen| \rightarrow_L examen[i] = -1)$
}

pred mismosEstudiantes($estudiantes1 : \mathsf{conj}\langle Estudiante\rangle, estudiantes2 : \mathsf{conj}\langle Estudiante\rangle$) {
$|estudiantes1| = |estudiantes2| \wedge$
$(\forall e : Estudiante\mathbb{Z})\ (e \in estudiantes1 \rightarrow e \in estudiantes2)$
}

pred esEstudianteCopiable($estudiante1 : Estudiante, estudiante2 : Estudiante$) {
$esCercano(estudiante1, estudiante2) \wedge$
$tieneUnEjercicioQueNoResolvi(estudiante1, estudiante2) \wedge$
$estudiante2.entregado = false$
}

pred esCercano($estudiante1 : Estudiante, estudiante2 : Estudiante$) {
$((estudiante1.posicion_0 = estudiante2.posicion_0) \wedge$
$(estudiante1.posicion_1 = estudiante2.posicion_1 + 2)) \vee$
$((estudiante1.posicion_0 = estudiante2.posicion_0) \wedge$
$(estudiante1.posicion_1 = estudiante2.posicion_1 - 2)) \vee$
$((estudiante1.posicion_0 = estudiante2.posicion_0 - 1) \wedge$
$(estudiante1.posicion_1 = estudiante2.posicion_1))$
}

pred tieneUnEjercicioQueNoResolvi($estudiante1 : Estudiante, estudiante2 : Estudiante$) {
$(\exists i : \mathbb{Z})\ (0 \leq i < |estudiante1.examen| \wedge_L$
$\neg\ resolvio(estudiante1.examen[i]) \wedge resolvio(estudiante2.examen[i]))$
}

pred resolvio($ejercicio : \mathbb{Z}$) { $0 \leq ejercicio \leq 9$ }

pred resolvioUnEjercicio($examenInicial : Examen, examenFinal : Examen$) {
$(|examenInicial| = |examenFinal|) \wedge_L$
$(\exists! i : \mathbb{Z})\ (0 \leq i < |examenInicial| \wedge_L \neg resolvio(examenInicial[i]) \wedge resolvio(examenFinal[i]) \wedge$

$(\forall j : \mathbb{Z})\ (0 \leq j < |examenInicial|\ \wedge\ j \neq i\ \rightarrow_L examenFinal[j] = examenInicial[j]))$
}

pred meCopio($examenInicial : Examen, examenFinal : Examen, examenCompañero : Examen$)
{
$(\exists i : \mathbb{Z})\ (0 \leq i < |examenInicial| \wedge_L \neg resolvio(examenInicial[i]) \wedge resolvio(examenCompañero[i]) \wedge$
$examenFinal[i] = examenCompañero[i])$
}

pred noSeModificaPosicionYEntregado($estudianteInicial : Estudiante, estudianteFinal : Estudiante$)
{
$estudianteFinal.posicion = estudianteInicial.posicion \wedge$
$estudianteFinal.entregado = estudianteInicial.entregado$

}

pred esSolucionDeExamen($examen1 : Examen, examen2 : Examen$) {
$|examen1| = |examen2| \land$
$solucionValida(examen1) \land$
$solucionValida(examen2)$
}

pred hayComoMuchoNexamenesDeLaDarkWeb($solucionDarkWeb : \text{seq}\langle\mathbb{Z}\rangle, N : \mathbb{Z}$,
$estudiantes : \text{conj}\langle Estudiante\rangle$) {
$cantidadDeAparicionesDelExamen(solucionDarkWeb, estudiantes) \leq N$
}

pred noTerminoExamen($estudiante : Estudiante$) {
$(\exists i : \mathbb{Z})\ (0 \leq i < |estudiante.examen|\ \land_L \neg resolvio(estudiante.examen[i])) \land$
$estudiante.entregado = false$
}

pred pasosValidos($pasos : \text{seq}\langle Examen\rangle, examenActual : Examen$) {
$(\forall i : \mathbb{Z})\ (0 \leq i < |pasos|\ \rightarrow_L\ |pasos[i]| = |examenActual|) \land$
$(\forall j : \mathbb{Z})\ (0 \leq j < |pasos[0]| \rightarrow_L pasos[0][j] = -1) \land$
$(\forall k : \mathbb{Z})\ (0 \leq k < |pasos| - 1 \rightarrow_L resolvioUnEjercicio(pasos[k], pasos[k+1])) \land$
$mismoExamen(pasos[|pasos| - 1], examenActual)$
}

pred mismoExamen($examen1 : Examen, examen2 : Examen$) {
$(|examen1| = |examen2|) \land_L$
$(\forall i : \mathbb{Z})\ (0 \leq i < |examen1|\ \rightarrow_L examen1[i] = examen2[i])$
}

pred noSeModificaExamenYPosicion($estudianteInicial : Estudiante, estudianteFinal : Estudiante$)
{
$estudianteFinal.posicion = estudianteInicial.posicion \land$
$estudianteFinal.examen = estudianteInicial.examen$
}

pred todosEntregaron($estudiantes : \text{conj}\langle Estudiante\rangle$) {
$(\forall e : Estudiante)\ (e \in estudiantes \rightarrow e.entregado = True)$
}

pred soySospechoso($e : Estudiante, estudiantes : \text{conj}\langle Estudiante\rangle$) {
$meCopieDeUnCercano(e, estudiantes) \lor$
$examenEsCopia(e.examen, estudiantes)$
}

pred meCopieDeUnCercano($e : Estudiante, estudiantes : \text{conj}\langle Estudiante\rangle$) {
$(\exists e1 : Estudiante)\ (e1 \in estudiantes \land esCercano(e, e1) \land$
$masDel60PorCientoIgual(e.examen, e1.examen))$
}

pred examenEsCopia($examen : Examen, estudiantes : \text{conj}\langle Estudiante\rangle$) {
$cantidadDeAparicionesDelExamen(examen, estudiantes) > 25PorCientoDeExamenes(estudiantes)$

}

pred masDel60PorCientoIgual($examen1 : Examen, examen2 : Examen$) {
$cantidadDeRespuestasIguales(examen1, examen2) > 60PorCientoDeRespuestas(examen1)$
}

pred hayEstudiantesRepetidos($estudiantes : \mathsf{seq}\langle Estudiante\rangle$) {
$(\exists i, j : \mathbb{Z})\ (0 \leq i, j < |estudiantes| \wedge i \neq j \wedge_L estudiantes[i] = estudiantes[j])$
}

pred sonNotasValidas($notas : Notas, estudiantes : \mathsf{conj}\langle Estudiante\rangle$) {
$(\forall i, j : \mathbb{Z})\ (0 \leq i, j < |notas| \wedge i \neq j \rightarrow_L$
$notas[i]_0 \neq notas[j]_0 \wedge$
$(\exists e : Estudiante)\ (e \in estudiantes \wedge notas[i]_0 = e.posicion))$
}

aux cantidadDeAparicionesDelExamen($examen : Examen, estudiantes : \mathsf{conj}\langle Estudiante\rangle$) : $\mathbb{R}$
$= \sum_{e \in estudiantes} IfThenElse(examen = e.examen, 1, 0)$

aux cantidadDeRespuestasIguales($examen1 : Examen, examen2 : Examen$) : $\mathbb{R} =$
$\sum_{i=0}^{|examen1|-1} IfThenElse(examen1[i] = examen2[i], 1, 0)$

aux 60PorCientoDeRespuestas($examen : Examen$) : $\mathbb{R} = (respuestasTotales(examen) * 60)/100$

aux respuestasTotales($solucion : Examen$) : $\mathbb{Z} = |solucion|$

aux 25PorCientoDeExamenes($estudiantes : \mathsf{conj}\langle Estudiante\rangle$) : $\mathbb{R} =$
(examenesTotales(estudiantes)*25)/100

aux examenesTotales($estudiantes : \mathsf{conj}\langle Estudiante\rangle$) : $\mathbb{R} = |estudiantes|$

aux nota($examen : Examen, solucion : Examen$) : $\mathbb{R} =$
(respuestasCorrectas(examen,solucion)/respuestasTotales(solucion))*10

aux respuestasCorrectas($examen : Examen, solucion : Examen$) : $\mathbb{Z} =$
$\sum_{i=0}^{|examen|-1} IfThenElse(examen[i] = solucion[i], 1, 0)$

aux cantidadDeSospechosos($estudiantes : conj\langle Estudiante\rangle$) : $\mathbb{R} =$
$\sum_{e \in estudiantes} IfThenElse(soySospechoso(e, estudiantes), 1, 0)$
}