Algunos comandos útiles para la resolución del parcial

Vectores (`vector`):

- `vector <int> v` crea un vector
- `v.size()` obtener el tamaño del vector
- `v.push_back(a)` insertar un elemento al final del vector
- `vector <int> v2 = v` crea una copia completa del vector

Pares (`pair`):

- `p = make_pair(a, b)` crea un par
- `p.first` accede al primer elemento
- `p.second` accede al segundo elemento

Enunciado

El origen del humor es algo desconocido. Se ha escrito sobre el origen del humor desde distintos enfoques[1]. En este caso no intentaremos detectar el origen, sino modelar su dinámica en un territorio dado, como podría ser un barrio.

Modelaremos un chiste: cómo viaja por el barrio, cuántas veces produce gracia, su ocaso y resurgimiento. Cada familia que vive en el barrio puede conocer o no el chiste, y después de un tiempo lo olvida. Cuando una familia escucha un chiste por primera vez, comienza un período en el cual el chiste les causa gracia. Al cabo de un tiempo, el chiste deja de ser gracioso. En este ejercicio asumiremos que el chiste dura como máximo 15 días hasta que pierde su gracia. Finalizado ese período, la familia olvida el chiste. Sin embargo, más adelante pueden volver a escuchar el chiste y reír nuevamente, en cuyo caso comienza un nuevo período en el cual el chiste es gracioso.

Representaremos el mapa de un barrio como una grilla, de tipo $seq\langle seq\langle \text{estado}\rangle\rangle$, donde cada celda de la grilla indica el **estado** en el que se encuentra la vivienda de una familia:

- El estado es un par, de tipo duraciónPeríodoActual × cantidadPeríodosAnteriores.
- La primera componente del estado es un entero que representa la **duración del período actual**, es decir, la cantidad de días que el chiste viene causándole *gracia* a una familia desde que empezaron a reírse del chiste por última vez. Si la familia está **aburrida**, es decir, no está actualmente riéndose del chiste, la duración del período actual es 0.
- La segunda componente del estado es un entero que representa la **cantidad de períodos anteriores**, es decir, la cantidad de períodos en total a lo largo de los cuales la familia se divirtió con el chiste. En este número no se contempla el período actual.

Los mapas serán de tamaño $N \times M$ siendo $N > 2$ y $M > 2$. Diremos que una familia es vecina de otra si se encuentra en la vivienda al norte, al sur, al este, o al oeste, en caso de existir. Por ejemplo:

Figura 1: En el mapa de la izquierda los vecinos de la familia en la posición marcada en rojo (0, 3), son las viviendas que se encuentran en las posiciones (0, 2) y (1, 3). En el mapa de la derecha los vecinos de la posición (2, 1) son (1, 1), (2, 0), (3, 1), (2, 2)

Todos los días se actualiza la situación del barrio usando las siguientes reglas:

- Si una casa está aburrida pero tienen un vecino que se está riendo del chiste, escuchan el chiste y la duraciónPeríodoActual pasa a ser 1.
- Todos los días se aumenta en 1 la duraciónPeríodoActual de aquellos que se están riendo (los que están aburridos permanecen en 0).
- Si una casa supera los 15 días riéndose, se aburren del chiste y dejan de reírse. La duraciónPeríodoActual pasa a ser 0. Además, la cantidadPeríodosAnteriores se incrementa en 1.

[1] El chistoso, de Isaac Asimov (1956).

Dados los siguientes renombres de tipos:

type posición $= \mathbb{Z} \times \mathbb{Z}$

type duraciónPeríodoActual $= \mathbb{Z}$

type cantidadPeríodosAnteriores $= \mathbb{Z}$

type estado = duraciónPeríodoActual $\times$ cantidadPeríodosAnteriores

type mapa $= seq\langle seq\langle$estado$\rangle\rangle$

Implementar las funciones enumeradas a continuación respetando la especificación.

Nota: Las funciones implementadas pueden ser reutilizadas en otros ejercicios. Para ello copiarlas de un ejercicio a otro.

Ejercicio 1.

```
bool vecinoRiendose(mapa m, posicion p)
```

Dado un mapa y una posicion devuelve si hay al menos una familia vecina que se está riendo del chiste.

proc vecinoRiendose (in m: mapa, in p: posición, out res: Bool) {
    Pre $\{mapaValido(m) \land_L posicionEnRango(p)\}$
    Post $\{res = True \leftrightarrow vecinoRiendo(m, p)\}$

    pred mapaValido (m: mapa) {
        $esMatrizDeMasDe2x2(m) \land_L posicionesValidas(m)$
    }
    pred esMatrizDeMasDe2x2 (m: mapa) {
        $|m| > 2 \land_L (|m[0]| > 2 \land (\forall f : \mathbb{Z})(0 < f < |m| \longrightarrow_L |m[f]| = |m[f-1]|))$
    }
    pred posicionesValidas (m: mapa) {
        $(\forall p :$ posición$)(posicionEnRango(m, p) \longrightarrow_L posicionValida(m, p))$
    }
    pred posicionValida (m: mapa, p: posición) {
        $0 \leq (m[p_0][p_1])_0 \leq 15 \land 0 \leq (m[p_0][p_1])_1$
    }
    pred posicionEnRango (m: mapa, p: posición) {
        $0 \leq p_0 < |m| \land 0 \leq p_1 < |m[0]|$
    }
    pred vecinoRiendo (m: mapa, p: posición) {
        $(\exists q :$ posición$)(esVecino(m, p, q) \land_L familiaRiendo(m, q))$
    }
    pred esVecino (m: mapa, p: posición, q: posición) {
        $posicionEnRango(m, q) \land$
        $((p_0 = q_0 + 1 \land p_1 = q_1) \lor$
        $(p_0 = q_0 - 1 \land p_1 = q_1) \lor$
        $(p_0 = q_0 \land p_1 = q_1 + 1) \lor$
        $(p_0 = q_0 \land p_1 = q_1 - 1))$
    }
    pred familiaRiendo (m: mapa, p:posición) {
        $0 < (m[p_0][p_1])_0 \leq 15$
    }
}

Ejercicio 2.

```
void situacionComica(mapa m, int &riendo, int &aburridos, int &nuncaEscucharon)
```

Dado un mapa queremos saber cuántas familias se están riendo actualmente, cuántas están aburridas (pero ya se rieron del chiste al menos una vez) y cuántas nunca escucharon jamás el chiste.

proc situacionComica (in m: mapa, out riendo: $\mathbb{Z}$, out aburridos: $\mathbb{Z}$, out nuncaEscucharon: $\mathbb{Z}$) {
    Pre $\{mapaValido(m)\}$
    Post $\{riendo = cantFamiliasRiendo(m) \land aburridas = cantFamiliasAburridas(m) \land$
    $nuncaEscucharon = cantFamiliasNuncaEscucharon(m)\}$

```
    aux cantFamiliasRiendo (m: mapa) : ℤ =
        ∑_{f=0}^{|m|-1} ∑_{c=0}^{|m[0]|-1} if familiaRiendo(m,(f,c)) then 1 else 0 fi;
    aux cantFamiliasAburridas (m: mapa) : ℤ =
        ∑_{f=0}^{|m|-1} ∑_{c=0}^{|m[0]|-1} if familiaAburrida(m,(f,c)) then 1 else 0 fi;
    aux cantFamiliasNuncaEscucharon (m: mapa) : ℤ =
        ∑_{f=0}^{|m|-1} ∑_{c=0}^{|m[0]|-1} if familiaNuncaEscucho(m,(f,c)) then 1 else 0 fi;
    pred familiaAburrida (m: mapa, p:posición) {
        (m[p0][p1])0 = 0 ∧ (m[p0][p1])1 > 0
    }
    pred familiaNuncaEscucho (m: mapa, p:posición) {
        (m[p0][p1])0 = 0 ∧ (m[p0][p1])1 = 0
    }
}
```

Ejercicio 3.

```
vector<posicion> mayorCantidadPeriodosAnteriores(mapa m)
```

Dado un mapa devuelve las posiciones de las familias que se ríeron durante la mayor cantidad de períodos.

```
proc mayorCantidadPeríodosAnteriores (in m: mapa, out res: seq⟨posición⟩) {
    Pre {mapaValido(m)}
    Post {|res| > 0 ∧_L cantidadesMaximas(m, res)}

    pred cantidadesMaximas (m: mapa, s:seq⟨posición⟩) {
        (∀p : posición)(posicionEnRango(m, p) ⟶_L
        (p ∈ s ↔ esCantMaximaPeriodosAnteriores(m, (m[p0][p1])1))))
    }
    pred esCantMaximaPeriodosAnteriores (m: mapa, max:int) {
        (∀p : posición)(posicionEnRango(m, p) ⟶_L (m[p0][p1])1 ≤ max)
    }
}
```

Ejercicio 4.

```
void actualizarMapa(mapa &m)
```

Dado un mapa queremos saber la evolución del chiste por el barrio cuando pasa un día. Se debe actualizar el mapa tomando en cuenta las reglas expresadas con anterioridad.

```
proc actualizarMapa (inout m: mapa) {
    Pre {m = m0 ∧ mapaValido(m0)}
    Post {|m| = |m0| ∧_L (∀p : posición)(posicionEnRango(m, p) ⟶_L actualizarEstadoFamilia(m0, m, p))}

    pred actualizarEstadoFamilia (m0: mapa, m: mapa, p: posición) {
        (familiaRiendo(m0, p) ∧ ¬terminaPeriodo(m0, p) ∧ m[p0][p1] = ((m0[p0][p1])0 + 1, (m0[p0][p1])1) ∨
        (familiaRiendo(m0, p) ∧ terminaPeriodo(m0[p0][p1]) ∧ m[p0][p1] = (0, (m0[p0][p1])1 + 1)) ∨
        (¬familiaRiendo(m0, p) ∧ vecinoRiendo(m0, (p0, p1)) ∧ m[p0][p1] = (1, (m0[p0][p1])1)) ∨
        (¬familiaRiendo(m0, p) ∧ ¬vecinoRiendo(m0, p) ∧ m[p0][p1] = m0[p0][p1]))
    }
    pred terminaPeriodo (m: mapa, p:posición) {
        (m[p0][p1])0 = 15
    }
}
```

Hint: Recuerde que m_0 representa el estado inicial del mapa mientras que m el valor que modificará, por lo que se sugiere no trabajar unicamente con m.