Universidad Nacional de Quilmes

Introducción a la Programación
2do Semestre 2025

Primer Parcial

- **EN LA PARTE SUPERIOR DE CADA HOJA PONER:**
  - **NOMBRE Y APELLIDO**
  - **DNI**
  - **COMISIÓN**
  - **NÚMERO DE HOJA y CANTIDAD TOTAL CON LA FORMA <número de hoja>/<cantidad total de hojas>**
- Leer el examen completo antes de empezar a resolver los ejercicios, ya que ayuda a comprender mejor el dominio. Se puede consultar cualquier material (en papel) que haya sido escrito antes de comenzar el examen y usar sin definir todas las funciones y procedimientos vistos durante la cursada.
- Pensar bien la estrategia a seguir, consultar lo que no se entienda y recordar que el parcial es una instancia más de aprendizaje donde algún docente va a dar una devolución personalizada de todo lo que se escriba. Es necesario aprovechar esta instancia como algo más que solamente un momento para sacar una nota.

Age of Gompire

"Age of Gompire" es un juego de estrategia en el que el jugador deberá ir consiguiendo recursos para aumentar su puntaje general en el juego. El juego ocurre en un mapa, que se divide en parcelas. En cada parcela puede haber una cierta cantidad de diferentes recursos, que incluyen, madera, piedra, oro y alimento. En Gobstones podemos representar cada parcela como una celda, y el mapa como la totalidad del tablero, y bolitas de diferentes colores para los recursos.

Las cantidades de los recursos no son fijas, y varían a lo largo del tiempo del juego, pudiendo aumentar o disminuir en base a diferentes criterios. Podemos en cualquier momento calcular la riqueza de un recurso, lo cual consiste en la suma de las cantidades de dicho recurso en la totalidad de las parcelas del mapa.

En esta evaluación se pide que implemente distintos procedimientos y funciones para conocer, gestionar y alterar los recursos del mapa. Se dispone de las siguientes operaciones primitivas para llevar adelante las tareas solicitadas:

```
recursoMadera()
PROPÓSITO: Describe el color con el que se
representa el recurso "madera".
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
recursoOro()
PROPÓSITO: Describe el color con el que se
representa el recurso "oro".
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
recursoPiedra()
PROPÓSITO: Describe el color con el que se
representa el recurso "piedra".
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
recursoAlimento()
PROPÓSITO: Describe el color con el que se
representa el recurso "alimento".
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
minRecurso()
PROPÓSITO: Describe el primero de los recursos.
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
maxRecurso()
PROPÓSITO: Describe el último de los recursos.
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
siguienteRecursoA_(recurso)
PROPÓSITO: Describe el recurso siguiente a
          **recurso**.
PARÁMETROS:- *recurso*: Color - recurso para el
cual se describe su siguiente.
TIPO: Color
PRECONDICIONES: Ninguna.
```

```
hayRecurso_Acá(recurso)
PROPÓSITO: Indica si hay unidades del
     recurso **recurso** en la parcela actual.
PRECONDICIÓN: Ninguna.
PARÁMETROS:- *recurso*: Color - color que
representa el recurso para el cual se indica si
hay en la parcela actual.
TIPO: Booleano.
```

```
IncrementarRecurso_Acá(recurso)
PROPÓSITO: Incrementar en 1 las unidades del
       recurso **recurso** en la parcela actual.
PARÁMETROS:- *recurso*: Color - color que
representa el recurso a incrementar.
PRECONDICIONES: Ninguna.
```

```
DecrementarRecurso_Acá(recurso)
PROPÓSITO: Decrementar en 1 la cantidad del
       recurso *recurso* en la parcela actual.
PARÁMETROS:- *recurso*: Color - color que
representa el recurso a decrementar
PRECONDICIONES:- Debe haber al menos una unidad
del recurso **recurso** en la parcela actual.
```

```
intensidadDelRecurso_Acá(recurso)
PROPÓSITO: Describe la cantidad de unidades del
recurso **recurso** de la parcela actual.
PARÁMETROS:- *recurso*: Color - color que
representa el recurso a determinar sus unidades.
PRECONDICIÓN: Ninguna.
PARÁMETROS:- *Debe haber al menos una unidad del
recurso **recurso** en la parcela actual.
```

Basados en este modelo y asumiendo en todos los casos que sobre el tablero hay un mapa de Age of Gompire bien representado, se pide:

Ejercicio 1)

El puntaje que obtiene un jugador tiene que ver con el concepto de "sector", que refiere a la parcela donde se ubica el cabezal (parcela actual) y todas las parcelas vecinas de manera ortogonal (hacia cada una de las direcciones). En ese sentido, los puntajes obtenidos tienen que ver con saber qué recursos hay disponibles en la totalidad del sector, siguiendo las reglas:

- Se obtienen 15 puntos si en el sector hay oro, o si hay alimento (solo 15 por cualquiera de los dos, no 15 por cada uno).
- Se obtienen 10 puntos si en el sector hay piedra, o si hay madera (solo 10 por cualquiera de los dos, no 15 por cada uno).
- Se obtienen 10 puntos adicionales si hay de los 4 tipos de recursos en el sector.

Se pide que escriba la función **puntosAObtenerEnSector** que describe los puntos obtenidos en el sector centrado en la parcela actual.

Ejercicio 2)

Para recompensar a los jugadores que han pasado varias horas de juego, "Age of Gompire" tiene el criterio de aumentar las cantidades del recurso de menor riqueza en una cantidad variable. Para modelar este comportamiento, se pide implementar el procedimiento **AumentarEn_RecursoDeMenorRiqueza** que, dado un número que representa la cantidad en la cual aumentar el recurso, aumenta cada parcela que contenga al menos una unidad del recurso de menor riqueza en todo el mapa tantas unidades como el número dado.

Ejercicio 3)

El juego también sigue criterios para penalizar a algunos jugadores, penalizando parcelas específicas. La forma de penalizar consiste en eliminar todas las unidades de oro que hubiera en la parcela, y restar 15 unidades de todos los otros recursos.

Considere la siguiente implementación de **PenalizarParcelaActual()**, y asuma la existencia del procedimiento ***DejarParcelaCon_Oro_Madera_PiedraY_Alimento*** que, dados cuatro números mayores o iguales a cero, y como su nombre sugiere, deja la parcela actual con exactamente tantas unidades de cada recurso como el número correspondiente dado.

```
procedure PenalizarParcelaActual() {
    /*
        PROPÓSITO: Penaliza la parcela actual.
        PRECONDICIONES: Ninguna.
    */
    madera := intensidadDelRecurso_Acá(recursoMadera()) - 15
    oro := 0
    alimento := intensidadDelRecurso_Acá(recursoAlimento()) - 15
    piedra := intensidadDelRecurso_Acá(recursoPiedra()) - 15
    DejarParcelaCon_Oro_Madera_PiedraY_Alimento(oro, madera, piedra, alimento)
}
```

Sabiendo esto se pide que

a) determine si la implementación propuesta es correcta (soluciona el problema planteado).
b) determina si la implementación es adecuada (sigue los buenos criterios trabajado en la materia).

JUSTIFICAR brevemente sus respuestas (no más de 5 renglones).