INF-134 Estructuras de datos 2024-1

Tarea 1: Jaque-mate

Profesores

Elizabeth Montero
elizabeth.montero@usm.cl

Andrés Navarro
andres.navarrog@usm.cl

Roberto Díaz
roberto.diazu@usm.cl

José Miguel Cazorla
jcazorla@usm.cl

Ricardo Salas
ricardo.salas@usm.cl

Ayudantes Casa Central

Sebastián Torrealba
sebastian.torrealba@usm.cl

Tomás Barros
tomas.barros@sansano.usm.cl

Bastián Jimenez
bjimenez@usm.cl

Franco Cerda
franco.cerda@usm.cl

Ayudantes San Joaquín

Carlos Lagos
carlos.lagosc@usm.cl

Gabriel Escalona
gabriel.escalona@alumnos.inf.utfsm.cl

Juan Alegria
juan.alegria@usm.cl

Lucas Morrison
lucas.morrison@sansano.usm.cl

Giuliana Zanetti
giuliana.zanetti@sansano.usm.cl

Reglas

- La presente tarea debe hacerse en *equipos de dos personas*. Toda excepción a esta regla está sujeta a autorización del ayudante *coordinador* Sebastián Torrealba.
- Debe usarse el lenguaje de programación **C++**. Al evaluarlas, las tareas serán compiladas usando el compilador **g++**, usando la línea de comando `g++ archivo.cpp -o output -Wall`. No se aceptarán variantes o implementaciones particulares de **g++**, como el usado por **MinGW** (normalmente ocupado en **Dev C++**). Se deben seguir los tutoriales disponibles en Aula USM.
- No se permite usar la biblioteca STL, así como ninguno de los contenedores y algoritmos definidos en ella (e.g. `vector`, `list`, etc.). Está permitido usar otras funciones de utilidad definidas en bibliotecas estándar, como por ejemplo `math.h`, `string`, `fstream`, `iostream`, etc.
- Recordar que una única tarea en el semestre puede tener nota menor a 30. El no cumplimiento de esta regla implica reprobación del curso.

Objetivos de aprendizaje

Comprender y familiarizarse con las estructuras y tipos de datos que provee C++.

- Paso de parámetros por valor y referencia
- Asignación de memoria dinámica
- Manejo de archivos

Además se fomentará el uso de las buenas prácticas y el orden en la programación del problema.

Problema a resolver: Jaque-mate

Contexto

Había una vez en un pequeño pueblo llamado Nlogonia, dos amigos, ayudantes de estructuras de datos, Sebastián y Ton solián jugar en sus tiempos libres.

Uno de esos dias de juego, Sebastián sugirió a Ton jugar ajedrez. ¿Qué es el ajedrez? preguntó Ton, a lo cual Sebastián responde:

*"El ajedrez es un juego de estrategia de dos jugadores que se juega en un tablero cuadrado dividido en **64** casillas de colores alternos (generalmente blanco y negro), dispuestas en un patrón de **8x8**. Cada jugador tiene un conjunto de dieciséis piezas, que incluyen un rey, una reina, dos torres, dos caballos, dos alfiles y ocho peones. El objetivo del juego es capturar al rey del oponente mientras se protege el propio, lo que se conoce como jaque-mate."*

Ton sorprendido con lo especifico de la respuesta de Sebastián se anima a jugar. Como Ton no sabe jugar ajedrez, Sebastián se aprovecha de su ingenuidad. Ton no sabe reconocer una situación de jaque-mate, en cambio Sebastián sí lo sabe y aprovecha esa información privilegiada.

Luego de que Ton perdiera tantas partidas decide pedirles a los estudiantes de estructuras de datos un programa en **C++** que le permita detectar si el tablero estaba en *jaque-mate*.

Problema a resolver

Se les dará como entrada un archivo de texto llamado `tablero.txt`. El archivo tendrá exactamente 9 líneas. La primera línea contendrá un número n que indicará la cantidad de piezas en el tablero. Cada una de las siguientes 8 líneas restantes tendrá exactamente 8 caracteres. *Con esto puedes asumir que siempre el tablero que se probará en la tarea será de* 8×8

Archivo que no está en jaque-mate

```
14
.T.....T
.C.P.P..
....P.P.
........
....A.X.
...P....
PP......
.KR.....
```

Archivo que está en jaque-mate

```
3
........
........
........
........
........
........
T.......
T......X
```

Nota: No necesariamente tiene que ser un tablero posible en la realidad. Se sabe que existen disposiciones imposibles, pero puede que se prueben casos donde el tablero es imposible de conseguir en un juego de ajedrez normal.

Si Ton se encuentra en *jaque-mate* se deberá responder: **Si**, en caso contrario se debe responder **No**. Se asegura que siempre existe una respuesta para cualquier caso de prueba. *Se pide explicitamente que solo respondan con "Si" o "No" para facilitar la correción por parte de los ayudantes.*

Un jugador se encuentra en *jaque-mate* cuando se cumplen las siguientes condiciones:

1. El rey del jugador está en *jaque*, lo que significa que una o más piezas del oponente amenazan con capturar al rey del jugador en el siguiente movimiento.

2. No hay ningún movimiento legal que el jugador pueda realizar para evitar que su rey sea capturado en el siguiente turno. Es decir el rey no puede moverse a una casilla adyacente que no esté amenazada por una pieza del oponente.

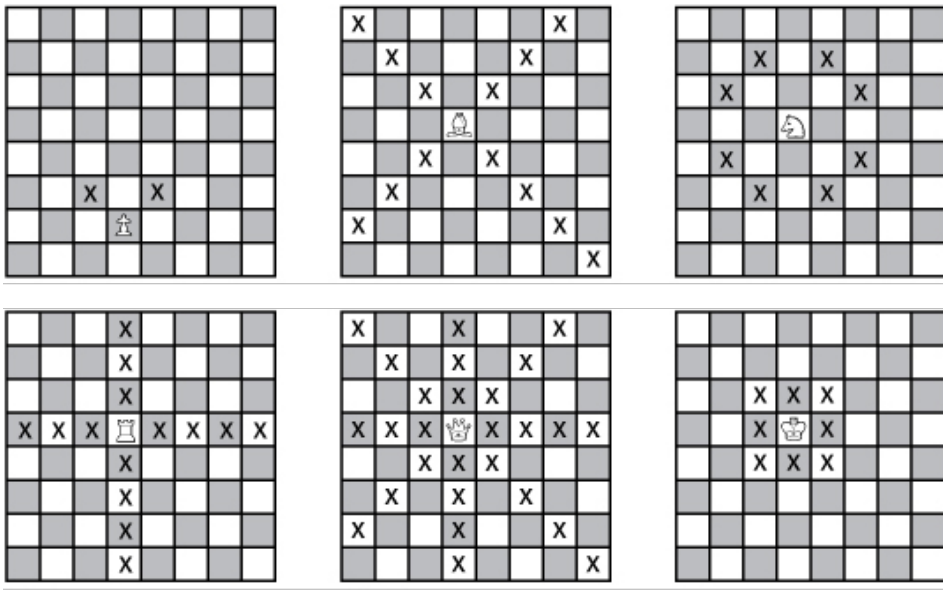
En otras palabras, el *jaque-mate* ocurre cuando el rey está bajo ataque y no hay forma de evitar su captura en el siguiente movimiento, sin importar qué jugada se realice.

Cada caracter dentro de cada línea del archivo `tablero.txt` tiene un significado distinto:

- **X:** Rey de Ton
- **K:** Rey de Sebastián
- **R:** Reina de Sebastián
- **P:** Peón de Sebastián
- **A:** Alfil de Sebastián
- **C:** Caballo de Sebastián
- **T:** Torre de Sebastián
- **. (punto):** Una casilla que no es ocupada por ninguna pieza

Los movimientos disponibles para el peón, alfil, caballo, torre, reina y rey de Sebastián son los siguientes, en el orden previamente mencionado de izquierda a derecha de arriba a abajo:

Las X definen que posiciones puede amenazar cada pieza



Requisitos al resolver el problema

Tendrás que implementar los siguientes registros y funciones para la tarea:

- **Registro Pieza:** Definirá los datos básicos para identificar una pieza

```
struct Pieza {
  char simbolo; // Define qué tipo de pieza es y su caracter
  int x, y;     // Su posición dentro del tablero [0, 7] x [0, 7]
};
```

- **Estructura Tablero:** Define un tablero con un arreglo de "Piezas"

```
struct Tablero {
  Pieza* piezas_tablero; // Lista de piezas que tiene el tablero
};
```

- **Función tableroEnJaqueMate:** Si está en *jaque-mate* retorna `true`, en caso contrario retorna `false`.

  IMPORTANTE: No deben hacer todo el código dentro de esta funcion, ustedes deben modularizar correctamente el jaque de cada pieza y abstraer la lógica para que quede un código lo suficientemente limpio y legible.

```
bool tableroEnJaqueMate(Tablero &tablero);
```

Entrega de la tarea

Entregue la tarea enviando un archivo comprimido **tarea1-apellido1-apellido2** (reemplazando sus apellidos según corresponda), en cuyo interior debe estar la tarea dentro de una carpeta también llamada **tarea1-apellido1-apellido2**, a la página *aula.usm.cl* del curso, el cual contenga:

- Los archivos con los códigos fuentes necesarios para el funcionamiento de la tarea. ¡Los archivos deben compilar!
- **nombres.txt**, Nombre, ROL, Paralelo y detalle de qué programó cada integrante del equipo.
- **README.txt**, Instrucciones de compilación en caso de ser necesarias, y la forma de compilación que usó (debe ser alguna de las indicadas en los tutoriales entregados en Aula USM).

Entrega mínima

La entrega mínima permite obtener una nota 30 si cumple con los siguientes requisitos:

- Programar al menos dos piezas de forma correcta
- Se manejan archivos (Se abre y cierra el archivo `tablero.txt`)

Si cumples con todos estos requisitos, no puedes tener una nota menor a 30.

Restricciones y consideraciones

- Se permite un único día de atraso en la entrega de la tarea. La nota máxima que se puede obtener en caso de entregar con un día de atraso es nota 50.
- Las tareas que no compilen no serán revisadas y tendrán que ser re-corregidas con el ayudante coordinador.
- Debe usar obligatoriamente alguna de las formas de compilación indicadas en los tutoriales entregados en Aula USM.
- Por cada Warning en la compilación se descontarán 5 puntos.
- Si se detecta COPIA la situación será revisada por ayudante y profesor coordinador.
- La prolijidad, orden y legibilidad del código fuente es obligatoria. Habrá descuentos si alguno de estos items no se cumple.

Directrices de programación

Las directrices corresponden a buenas prácticas de programación, las cuales serán tomadas en consideración para la evaluación de la tarea. El código fuente del programa debe estar estructurado adecuadamente en archivos (separados de ser necesario). Si el código fuente está desordenado, se pueden descontar hasta 20 puntos de la nota. Cada función programada debe tener comentarios de la siguiente forma:

```
/*****
* TipoFunción NombreFunción
******
* Resumen Función
******
* Input:
* tipoParámetro NombreParámetro : Descripción Parámetro
* .......
******
* Returns:
* TipoRetorno, Descripción retorno
*****/
```

- Por cada comentario faltante, se restarán 5 puntos.
- Por último, la indentación (1 TAB o 4 espacios), es muy importante. Por cada bloque mal indentado, se quitarán 10 puntos.