Assignment 2

Note: For this assignment you are given **3 choices** and you can pick **any one of them.** All the choices have same difficulty level and marking criteria. You are advised to read all the three choices and choose the one that interests you.

Submission: Make a *private* GitHub repository for your assignment and add meharfatimakhan as collaborator. Also submit your GitHub repository link here: https://docs.google.com/forms/d/e/1FAIpQLSdvenAiFetxV3IIeJp14SJAWaepCrLKUamI8yRgKcBm-wPpIQ/viewform

Deadline: 20th March, 2020 till 11:55 PM

Choice 1

Ultimate Tic-Tac-Toe

We all know Tic-Tac-Toe ends in a predictable draw when both players choose optimal moves. To make the game interesting, a group of mathematicians decided to fill in each square of the board with a smaller Tic-Tac-Toe board to produce what is now called **Ultimate Tic-Tac-Toe**.

Challenge:
Here is an abstract flow of the application:

1. The program asks for the name of player.

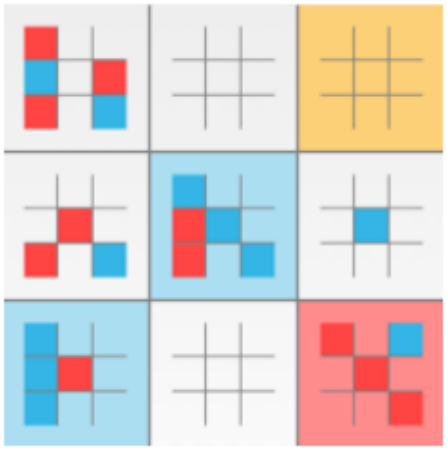
2. Player is asked to choose the preferable sign in the game. Computer takes the opposite sign to that.

3. The game starts with the empty ultimate tic-tac-toe board (shown below)

4. You need to win the game by continually printing the best next move, based on the game rules below. You will lose the test case if you write to an invalid board and/or write to a square that already has an X or O.

For the purposes of these instructions:

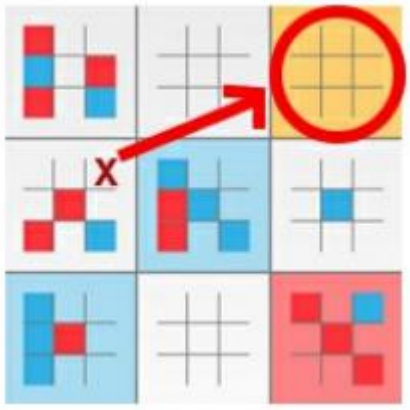
- **board** = one of the 9 tic-tac-toe game boards
- **square** = one of the 81 individual squares



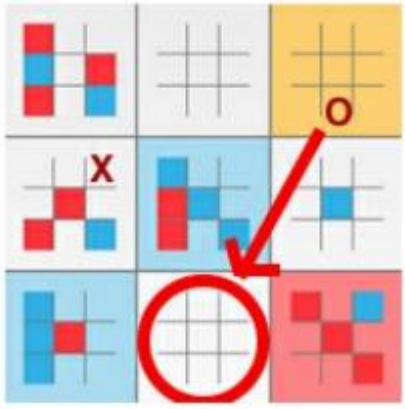
Game Rules

Like the original Tic-Tac-Toe, Player 1 is represented by either 'X' or ‘O’ and Player 2 is represented by the opposite sign (if Player 1 is ‘X’, then Player 2 is ‘O’). To start the game, Player 1 places his/her chosen sign on any one of the 81 empty squares, and then players alternate turns.

However, after the initial move, players must play the board that mirrors the square from the previous player.



For example: If Player 1 places an X in the upper-right *square* of a board, then Player 2 must play the upper-right *board*.



Continuing the example: If Player 2 places an O on the lower-middle square, then Player 1 must next play the lower-middle board.

Here are the necessary exceptions:

If the next move is to a board that is full or has already been won, then that player may choose an **open** square on **any** board, for that turn.
If a player marks 3 consecutive squares (horizontally, vertically or diagonally) on any given board, he wins that board. The first player to win 3 consecutive *boards* (horizontally, vertically or diagonally) wins the game.

NOTE:

- **You don't get to pick which of the nine boards to play on.** That's determined by your opponent's previous move. **Whichever *square* he picks, that's the *board* you must play in next.**

- What if one of the small boards results in a tie? Then the board counts for **neither** X nor O.

- Maintain a score of each player in a database.

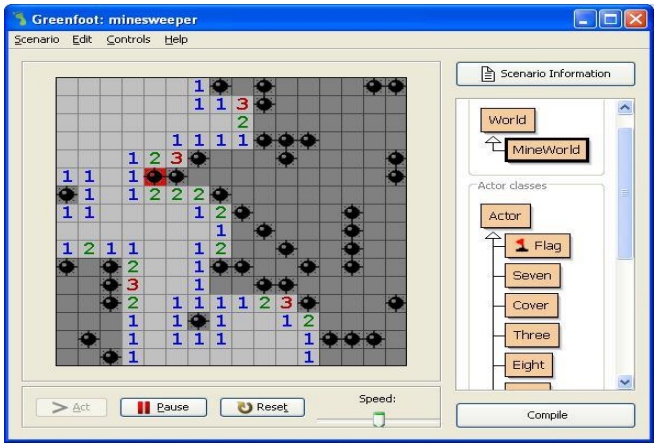
- There should also be a warning notifying the user if he/she makes wrong move by putting the sign into an already marked cell.

- You have to build **GUI (Graphical User Interface) using JAVA Swing package for this game. Be as creative as you can. You will be marked on the quality of the GUI separately!**

Choice 2

Minesweeper

Minesweeper is a computer game that became popular in the 1980s, and is still included in some versions of the Microsoft Windows operating system. This problem has a similar idea, but it does not assume you have played Minesweeper.



The goal of the game is to sweep all mines from a mine field. If the player clicks on the cell which contains a mine, the mine detonates and the game is over.

The Minesweeper game is controlled solely by mouse. We react to left and right mouse clicks.

Further a cell can contain a number or it can be blank. The number indicates how many mines are adjacent to this particular cell. We set a mark (red flag) on a cell by right clicking on it. This way we indicate, that we believe, there is a mine. To remove a flag, right click on it.

There are 15 images used in this game:

A cell can be surrounded by maximum of 8 mines, so we need numbers 1, 2, 3, 4, 5, 6, 7, 8. We need images for an empty cell, a mine, a detonated mine (game is over), a marked cell (flag), a covered cell and finally for a wrongly marked cell.

Twist:
Our variant works very similarly to standard Minesweeper, but with multiple players simultaneously playing on a single board. In both versions, players lose when they try to dig an untouched square that happens to contain a bomb. Whereas a standard Minesweeper game would end at this point, in our version, the game keeps going for the other players. You have to allow as many users to play the game as possible. In our version, when one player blows up a bomb, they still lose, and the game ends for them (their connection ends), but the other players may continue playing. The square where the bomb was blown up is now a *dug* square with no bomb for the other players. You have to achieve this task through **multithreading**.

Challenge:

- You have to build **GUI (Graphical User Interface) using JAVA Swing package for this game. Be as creative as you can. You will be marked on the quality of the GUI separately!**
- Each player will be playing against the computer
- Maintain a database of the scores achieved of each player.
- The high score will be measured w.r.t to time. Time is the time taken to open all cells without mines. Let's say 3 players are playing the game simultaneously. If player 1 completes the earliest, then game shall display his time and name, display it in another window stating that player 1 is the winner. If player 2 completes after player 1 then display his time and name below player 1's. Similarly display player 3's name and time if he completes after player 2. In case all the players have opened no mine (they are safe), you should let them all play till the end.
- If a player opens a mine, let's say player 3 opens a mine then you should end that player's thread and display it in the window that player 3 is out.
- Make the game generic. At the start of the game, ask the number of players that are playing and also prompt the user to enter the value of 'N' which will create an NxN board.