Name: Andrew Nyaisonga
Using my friend Java

Running the Project:

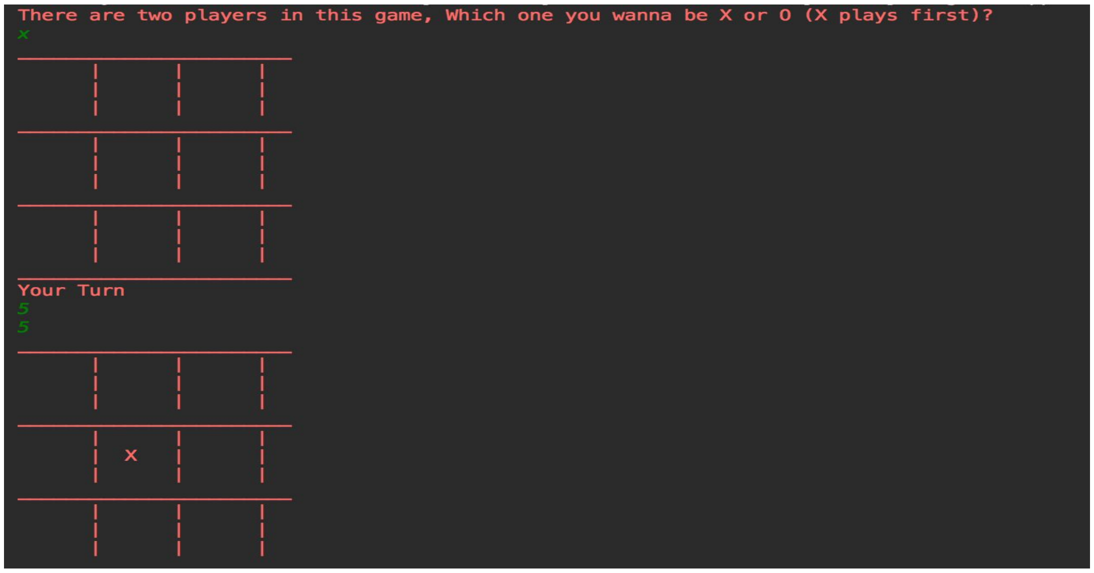
The project can be runned on terminal and it takes standard input

Part 1:

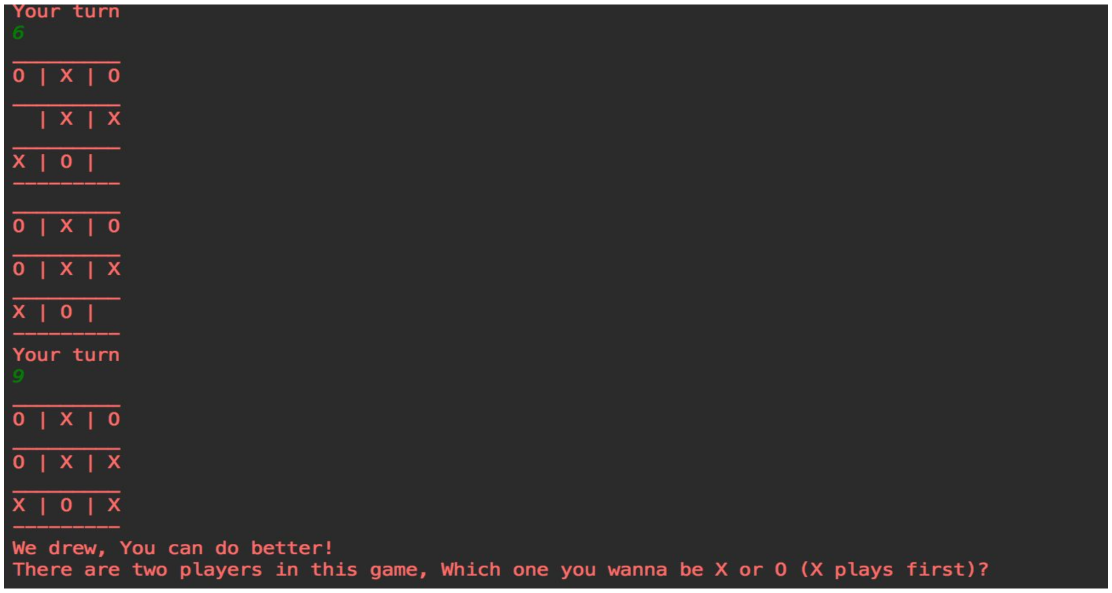
On the basic TTT, I used a simple Minimax method to find the best move for every state.
The program uses tree structure uses minimax for each possible state to find the best possible move.Because TTT is very easy and has few states (less than 9!) we don't have to worry on estimating the cost of each state. In fact in this program the computer will never lose.

The state of the game will be printed using the standard output after each move. The program takes no time at all to make a move. The computer will play as the 1 and the user will be 2 in the background. This still allows the user to choose to be X or O. The game always start at player turn X as specified by the project requirements. All the states are represented by an array of length 9. The template of the methods has be taken from the https://playtictactoe.org/ game as I looked to make realistic game. Template as the type of method I might need and naming methods although they didn't use minimax to allow other players to win.

The whole of part one contain 3 classes named State, Basic_T_Model and TTTBasic_Main indicating there are from the Basic part of the project. The Minimax method (called MINIMAX on my program) is on the Basic_T_Model. The minimax works the same way, it searches using DFS to the all the move and chooses the best one. It uses MAX and MIN helper function to decide the best possible value the computer can get at that state using recursion. It does this by passing the state and finding all the possible actions using the getapplicableMove that returns all the empty tiles and check which one will be best for the min values and do over till it gets to the terminal state.As described the player can just enter number from 1-9 through standard input. If the position is taken, it will request another move from the player.



```
There are two players in this game, Which one you wanna be X or O (X plays first)?
x
_______________________
      |      |      |
      |      |      |
      |      |      |
_______________________
      |      |      |
      |      |      |
      |      |      |
_______________________
      |      |      |
      |      |      |
      |      |      |
_______________________
Your Turn
5
5
_______________________
      |      |      |
      |      |      |
      |      |      |
_______________________
      |      |      |
      |  x   |      |
      |      |      |
_______________________
      |      |      |
      |      |      |
      |      |      |
```



```
Your turn
6
_________
O | X | O
_________
  | X | X
_________
X | O |
---------
_________
O | X | O
_________
O | X | X
_________
X | O |
---------
Your turn
9
_________
O | X | O
_________
O | X | X
_________
X | O | X
---------
We drew, You can do better!
There are two players in this game, Which one you wanna be X or O (X plays first)?
```

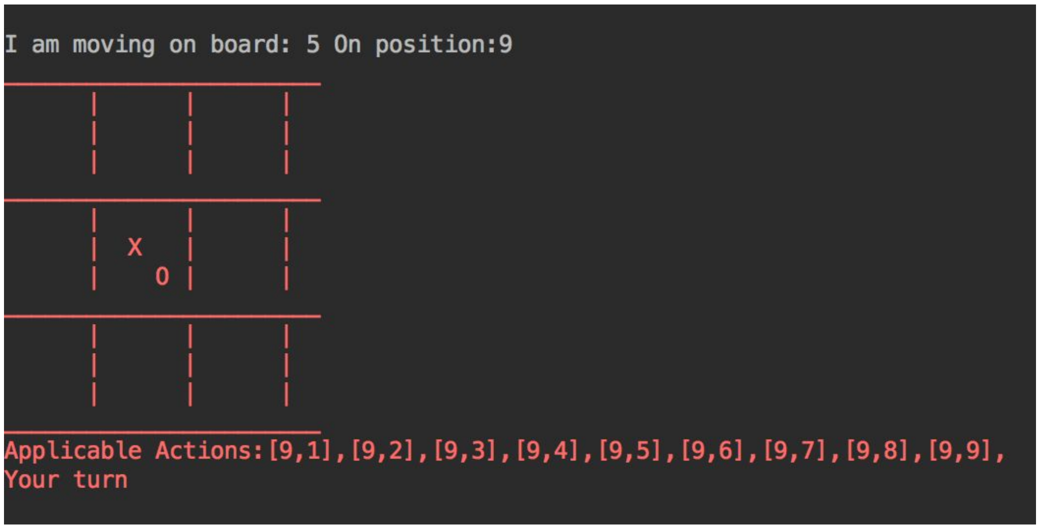
Pictures from the actual game

Part 2

Very similar to part 1 except this uses 9 board instead of one. To handle this slight change I used array of array. Each index will correspond to the board and within that board you will get the representation of the first TTT.

It contain 4 classes TTT9Board_main,State_Advance,Advance_T_Model and Heuristic_Fuction. The Heuristic function uses the value of the number of row, column and diagonal that the computer can make to move minus that number of rows, column and diagonal moves the user can make to win. This will help estimate the cost of each

state across all the possible options, it's pretty cool heuristic. I used Alpha-Beta algorithm to prune out the tree and help in increasing the speed.
Talking about speed, after playing a lot of games with http://www.stratigery.com/gen9.html. It was pretty clear to me in order to have a good speed and an entertaining game I have to limit my search to depth of 6. As seen on function H_MINIMAX the depth is limited to 6 on which we will return the Heuristic_Function estimate. You can always make the game more interactive by asking the player to specify which level they want and use that as the depth to be evaluated.



```
I am moving on board: 5 On position:9
```

```
Applicable Actions:[9,1],[9,2],[9,3],[9,4],[9,5],[9,6],[9,7],[9,8],[9,9],
Your turn
```

Pictures from the actual game

What could have been improved

1. Handling of exceptions, I would have loved to that but it was not part of the project and it really doesn't test anything you know
2. Heuristic function is hard to come up with. Without the help I wouldn't have been able to find one and according to the internet, this might not be optimal
3. I would have loved to make GUI for this program and I think I will later in the future.

Reference

1. http://www.stratigery.com/gen9.html for understanding the 9 board game and giving a good estimate for the state
2. https://playtictactoe.org/ for the layout and good function naming for some of the classes