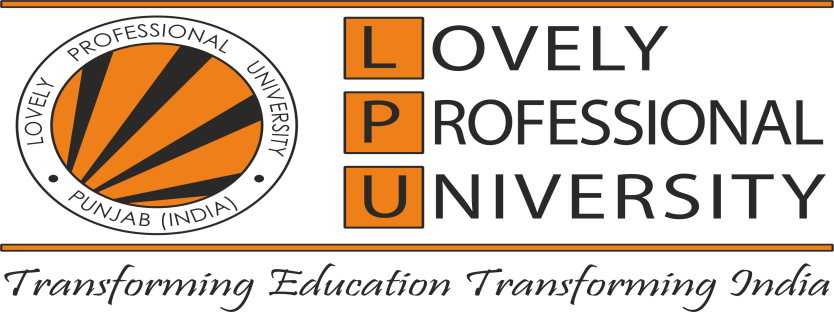
**A SIMPLE GAME IN AI**

**TIC-TAC-TOE**

**Course code: INT 404**

**Section: K18FR**

|  |  |  |
| --- | --- | --- |
| **Registration No.** | **Name of Students** | **Roll No.** |
| 11811816 | **NEHA ARORA** | RK18FRB64 |
| 11806110 | **RISHABH JAISWAL** | RK18FRB63 |
| 11805134 | **B.SATYAM RANA** | RK18FRB62 |
| 11805272 | **VADDEPALLY JAYAPRAKASH NARAYANA** | RK18FRB61 |

****

**Submitted to:-**

**Name of the faculty: Ms. Ishatpreet kaur mam**

**Department of Intelligent Systems**

**School of Computer Science Engineering**

**Lovely Professional University, Jalandhar**

APRIL-2020

**Student Declaration**

This is to declare that this report has been written by us. No part of the report is copied from other sources. All information included from other sources has been duly acknowledged. We aver that if any part of the report is found to be copied, we are shall take full responsibility for it.

NEHA ARORA

11811816

64

RISHABH JAISWAL

11806110

63

B.SATYAM RANA

11805134

62

VADDEPALLY JAYAPRAKASH NARAYANA

11805272

61

Place : LOVELY PROFESSIONAL UNIVERSITY,PHAGWARA,PUNJAB

Date : 14-04-2020

**TABLE OF CONTENTS**

**TITLE PAGE NO.**

1. **Introduction 1**
2. **Distribution of module 2**
3. **Flow chart 3**
4. **Representing the board as data 4**
5. **Algorithm 5-6**
6. **Test cases 7-10**
7. **Player wins computer loses 7**
8. **Player loses computer wins 8-9**
9. **Draw match 10**
10. **Gantt chart 11**
11. **Summary 12**

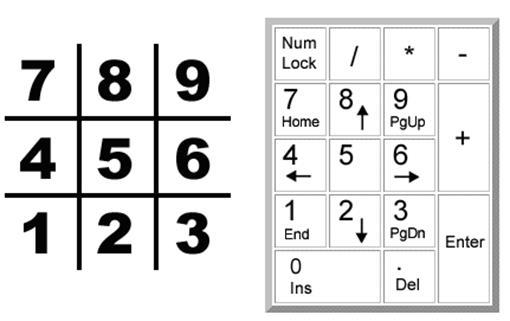
**REPOSITORY LINK:** **https://github.com/NehaArora9/AI-PROJECT.git**

**INTRODUCTION**

It is a **TIC-TAC-TOE** game against a simple Artificial Intelligence. An **artificial intelligence (or AI)** is a computer program that can intelligently respond to the player’s moves. This game doesn’t introduce any complicated new concepts. The artificial intelligence that plays Tic-Tac-Toe is really just a few lines of code.

Two people play Tic-Tac-Toe with paper and pencil. One player is X and the other player is O. Players take turns placing their X or O. If a player gets three of their marks on the board in a row, column or one of the two diagonals, they win. When the board fills up with neither player winning, the game ends in a draw. This project doesn’t introduce any new programming concepts. It makes use of our existing programming knowledge to make an intelligent Tic-Tac-Toe player.

Let’s get started by looking at a sample run of the program. The player makes their move by entering the number of the space they want to go. These numbers are in the same places as the number keys on your keyboard's keypad.

  
**The board is numbered like the keyboard's number pad.**

**DISTRIBUTION OF MODULE**

**NEHA ARORA**

* Making report file
* Algorithm
* Design part of project
* Code implementation

**RISHABH JAISWAL**

* Main module

**B.SATYAM RANA**

* Player move module

**VADDEPALLY JAYAPRAKASH NARAYANA**

* Flow chart
* Computer move module
* Printboard module

**FLOW CHART**

**(Graphical Results)**

A flow chart of Tic-Tac-Toe could look like. In the Tic-Tac-Toe computer program the player chooses if they want to be X or O. Who takes the first turn is randomly chosen. Then the player and computer take turns making moves.

The boxes on the left side of the flow chart are what happens during the player’s turn. The right side shows what happens on the computer's turn.

After the player or computer makes a move, the program checks if they won or caused a tie, and then the game switches turns. After the game is over, the program asks the player if they want to play again.



**Flow chart for Tic Tac Toe**

**REPRESENTING THE BOARD AS DATA**

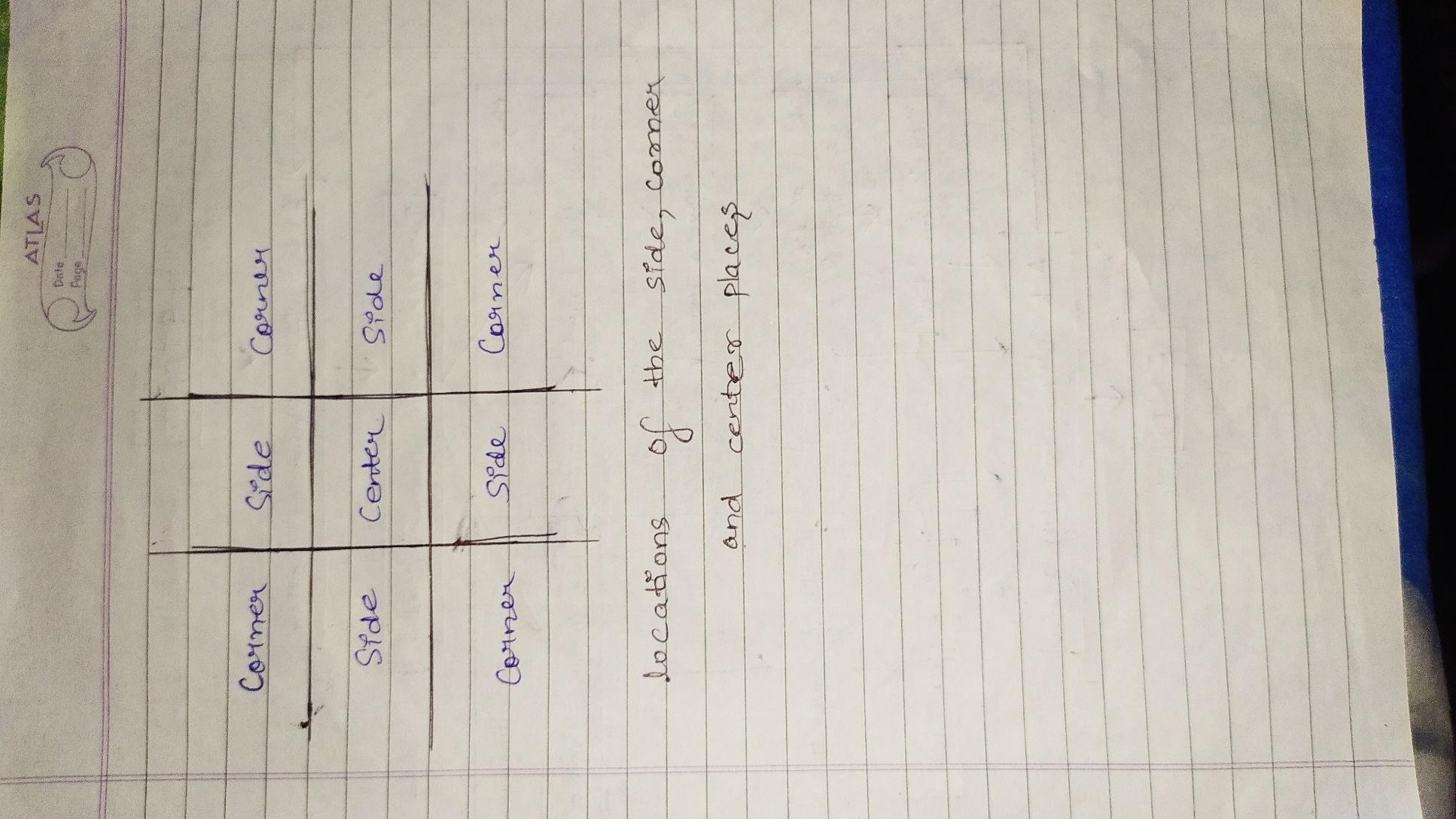
First, you must figure out how to represent the board as data in a variable. On paper, the Tic-Tac-Toe board is drawn as a pair of horizontal lines and a pair of vertical lines, with either an X, O, or empty space in each of the nine spaces.

In the program, the Tic-Tac-Toe board is represented as a list of strings. Each string will represent one of the nine spaces on the board. To make it easier to remember which index in the list is for which space, they will mirror the numbers on a keyboard’s number keypad.

The strings will either be **'X' for the X player, 'O' for the O player, or a single space ' ' for a blank space.**

The player will enter a number from 1 to 9 to tell the game which space they want to move on.

We will label three types of spaces on the Tic-Tac-Toe board: corners, sides, and the center.



**Locations of the side, corner, and center places**

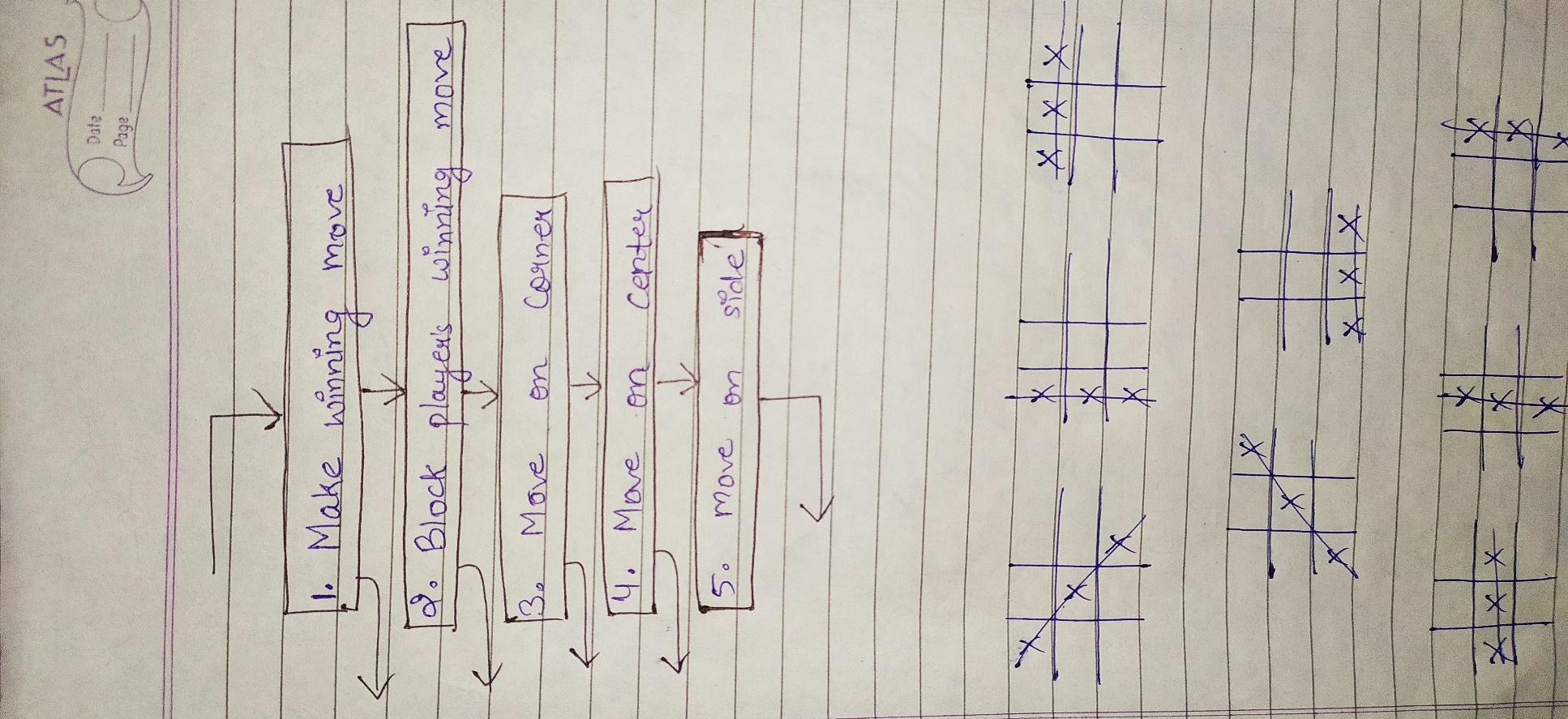
**ALGORITHM**

An **algorithm** is a finite series of instructions to compute a result. A single program can make use of several different algorithms. An algorithm can be represented with a flow chart. The Tic-Tac-Toe AI’s algorithm will compute the best move to make.

The AI smarts for playing Tic-Tac-Toe will follow a simple algorithm.

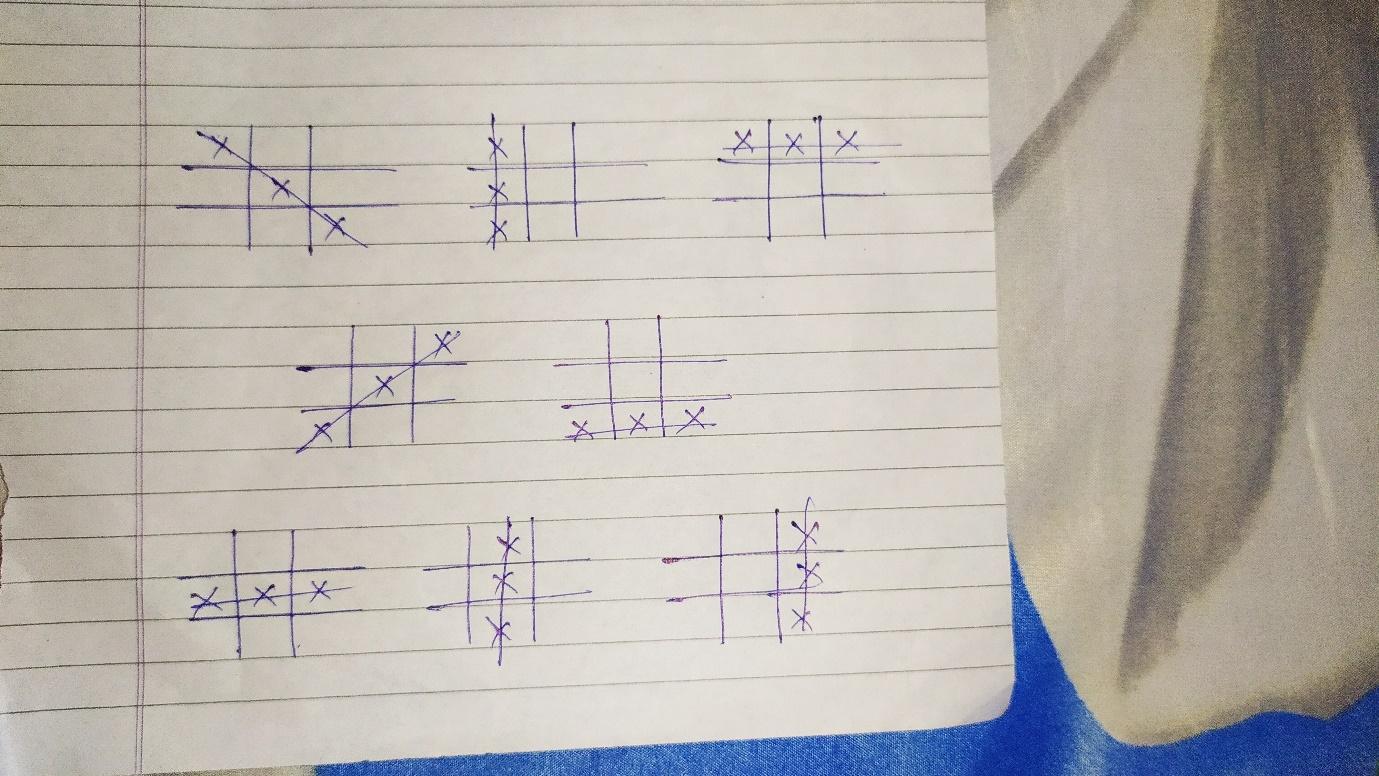
1. First, see if there’s a move the computer can make that will win the game. If there is, take that move. Otherwise, go to step 2.
2. See if there’s a move the player can make that will cause the computer to lose the game. If there is, move there to block the player. Otherwise, go to step 3.
3. Check if any of the corner spaces (spaces 1, 3, 7, or 9) are free. If so, move there. If no corner piece is free, then go to step 4.
4. Check if the center is free. If so, move there. If it isn’t, then go to step 5.
5. Move on any of the side pieces (spaces 2, 4, 6, or 8). There are no more steps, because if the execution reaches step 5 the side spaces are the only spaces left.

This all takes place in the **“Get computer's move”** box on the flow chart.



**Algorithm of tic-tac-toe game**

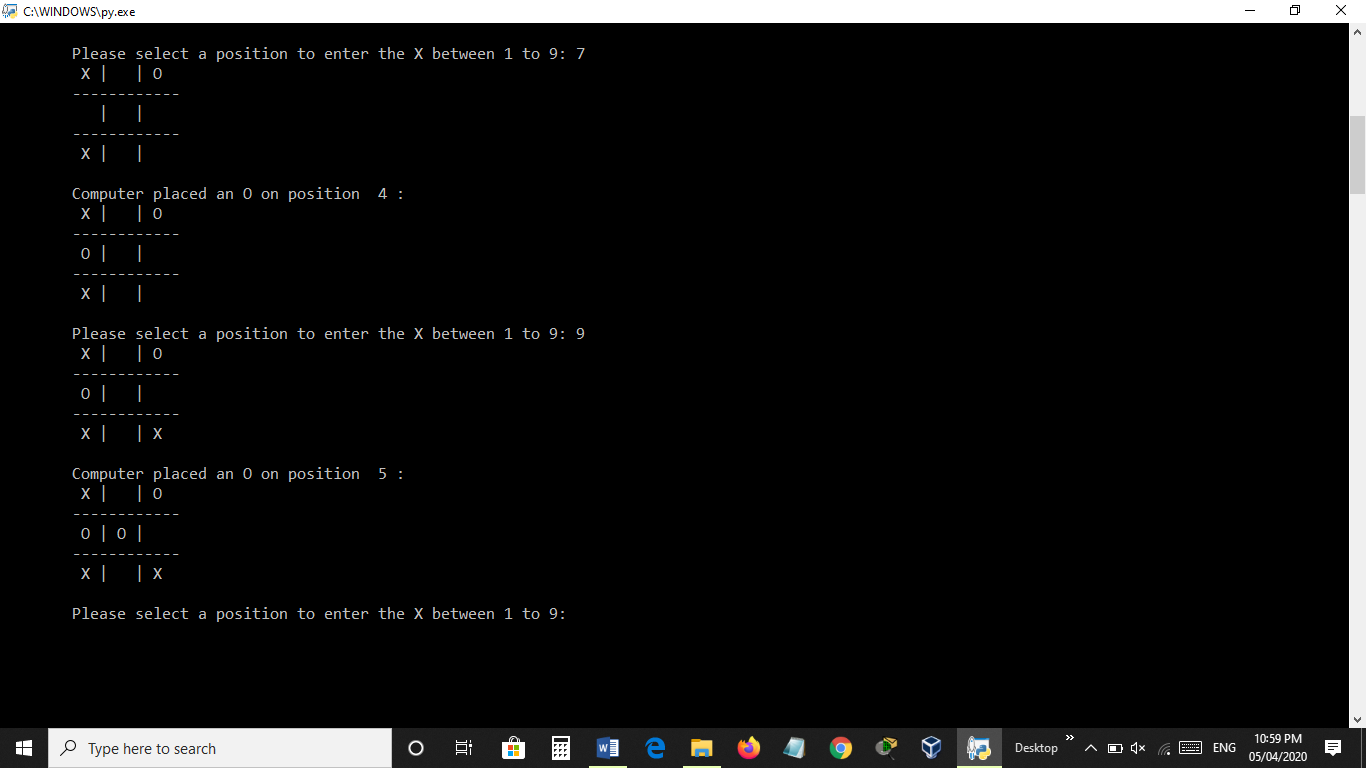
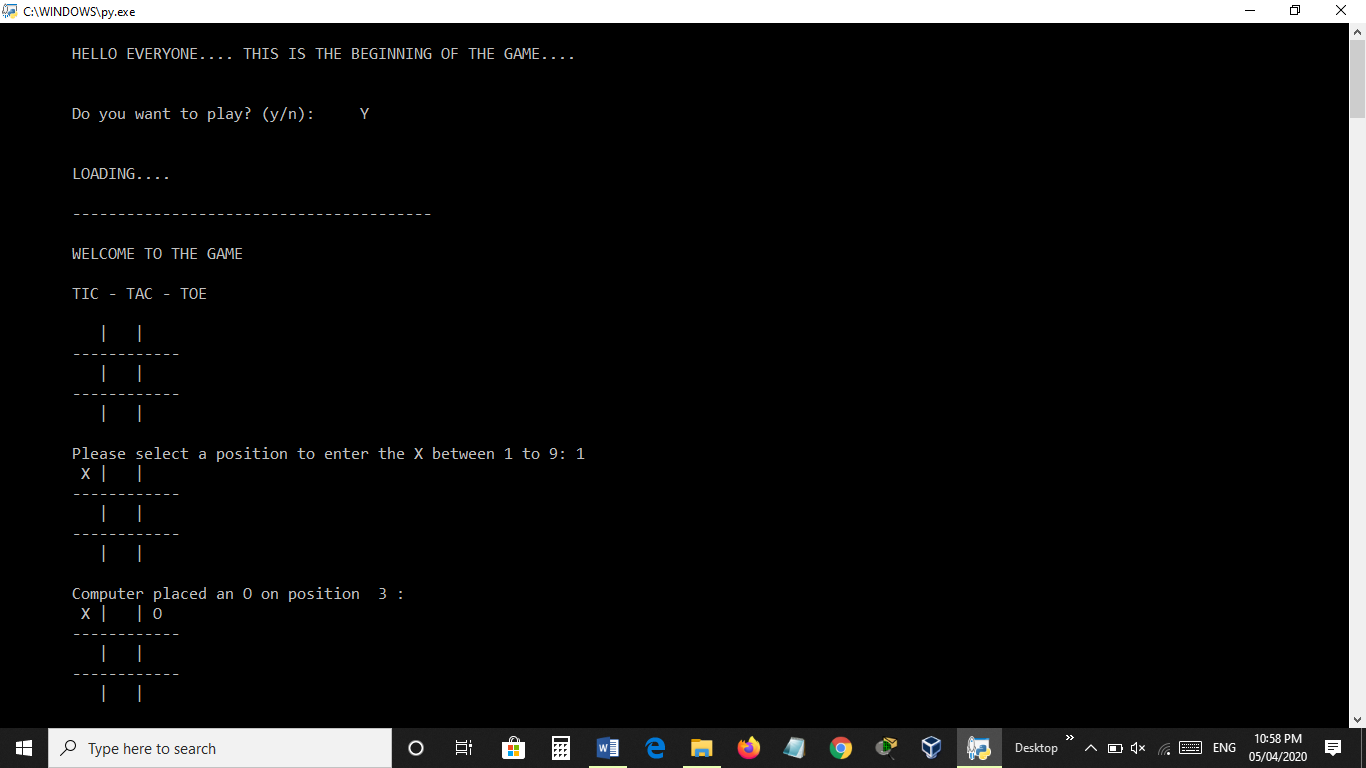
There are eight possible ways to win at Tic-Tac-Toe. You can have a line across the top, middle, and bottom rows. Or you can have a line down the left, middle, or right columns. Or you can have a line over either of the two diagonals. This means only one of the eight ways must be true in order for us to say that the player who owns letter is the winner.

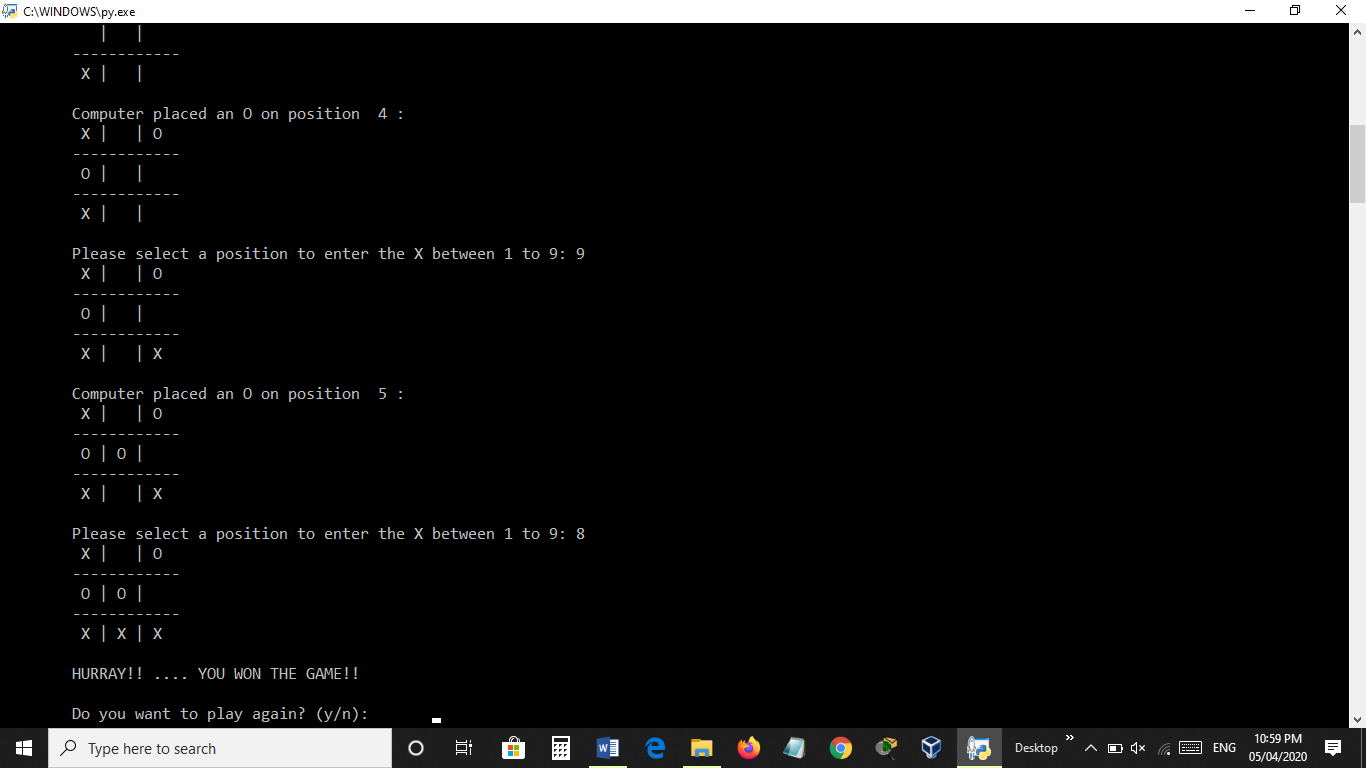


**8 possible ways to win the tic-tac-toe game**

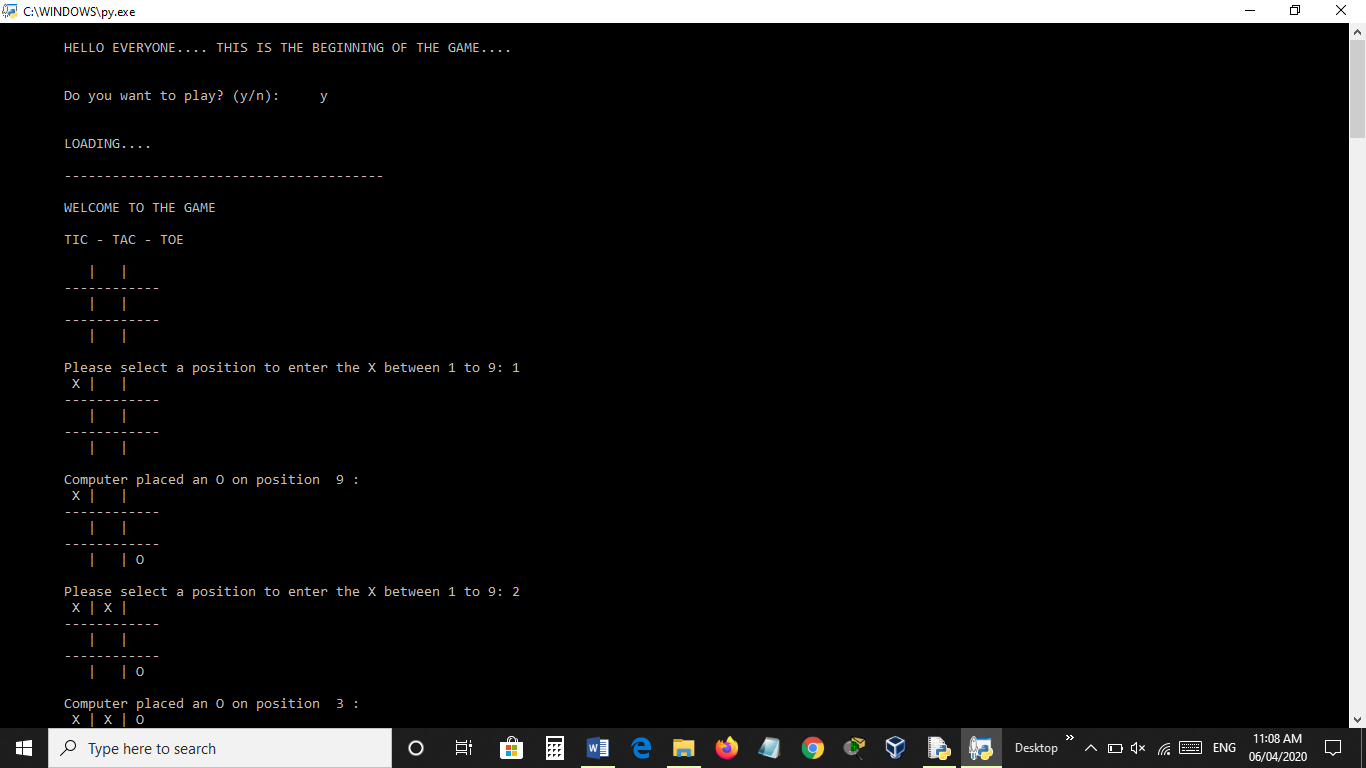
**TEST CASES**

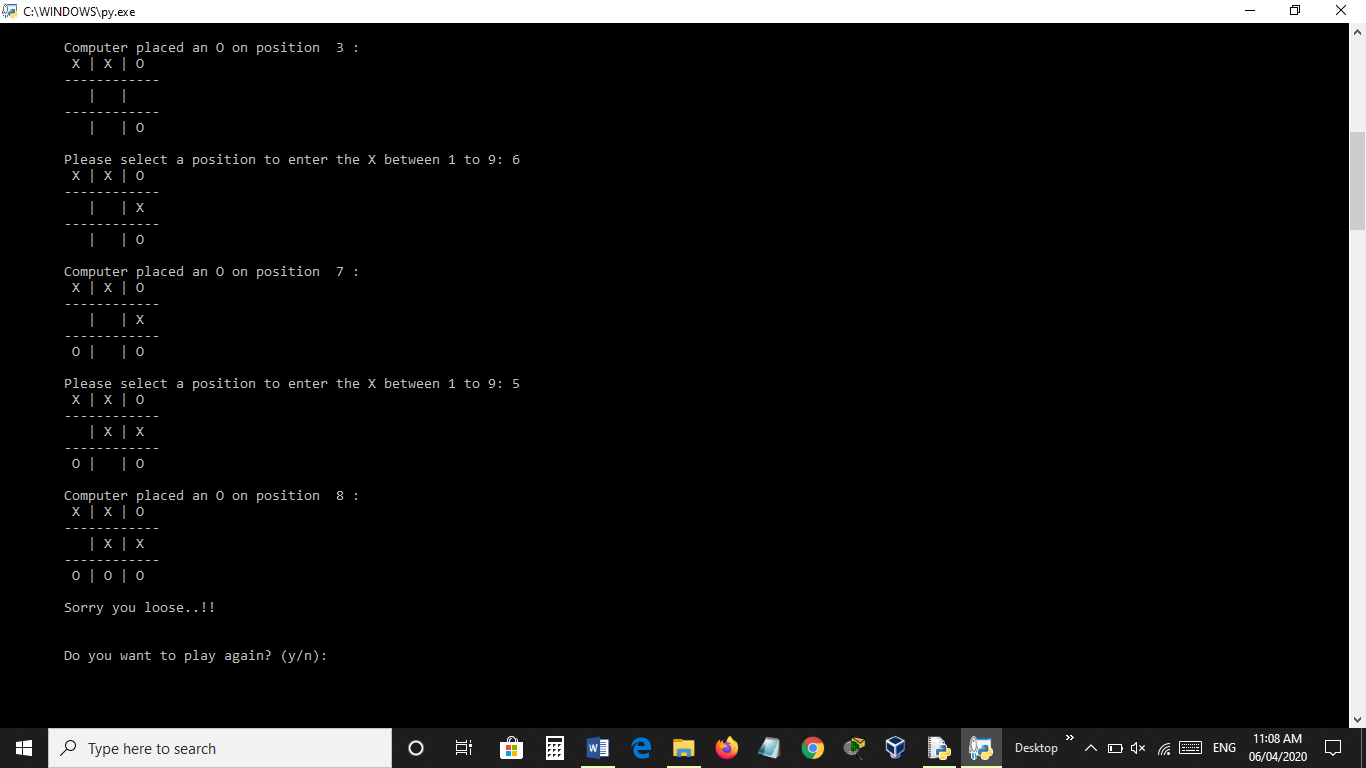
**1) WHEN PLAYER ‘X’ WINS AND COMPUTER ‘O’ LOSES -**



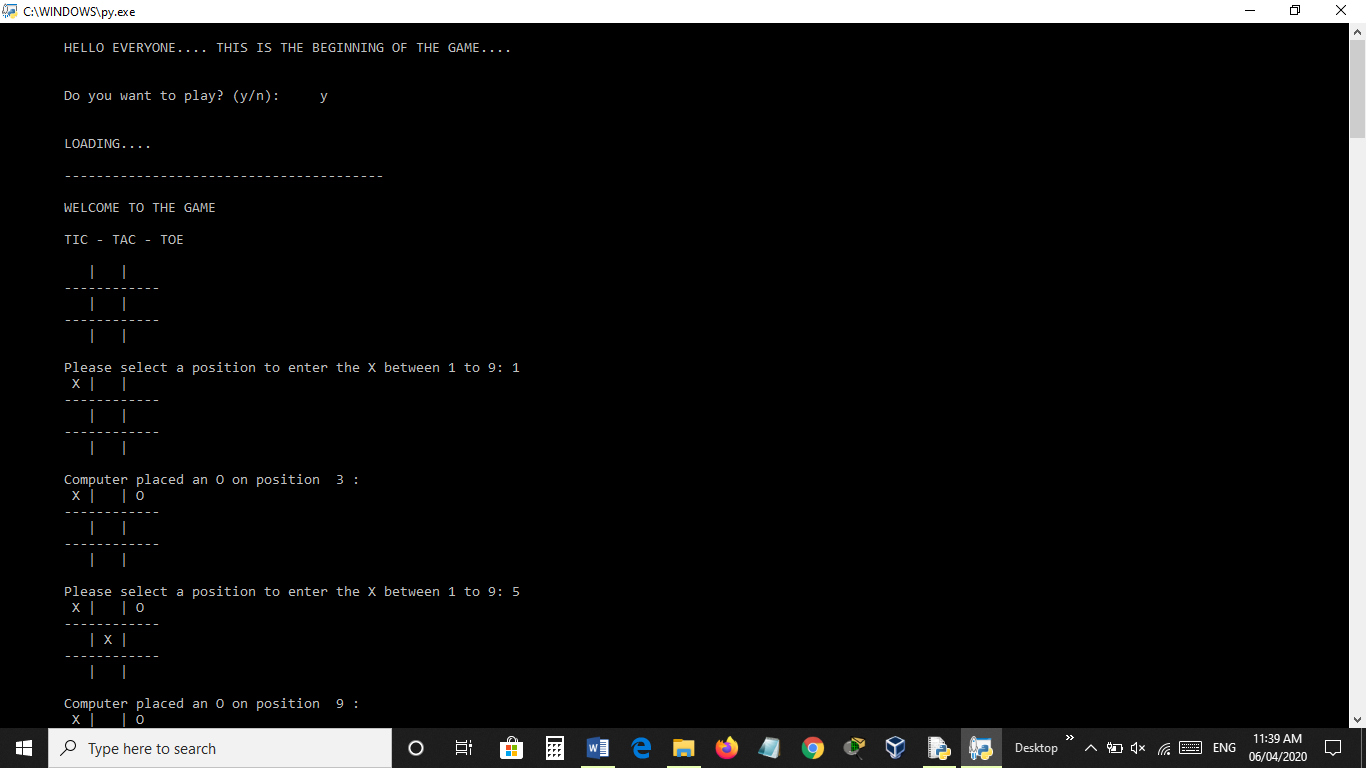
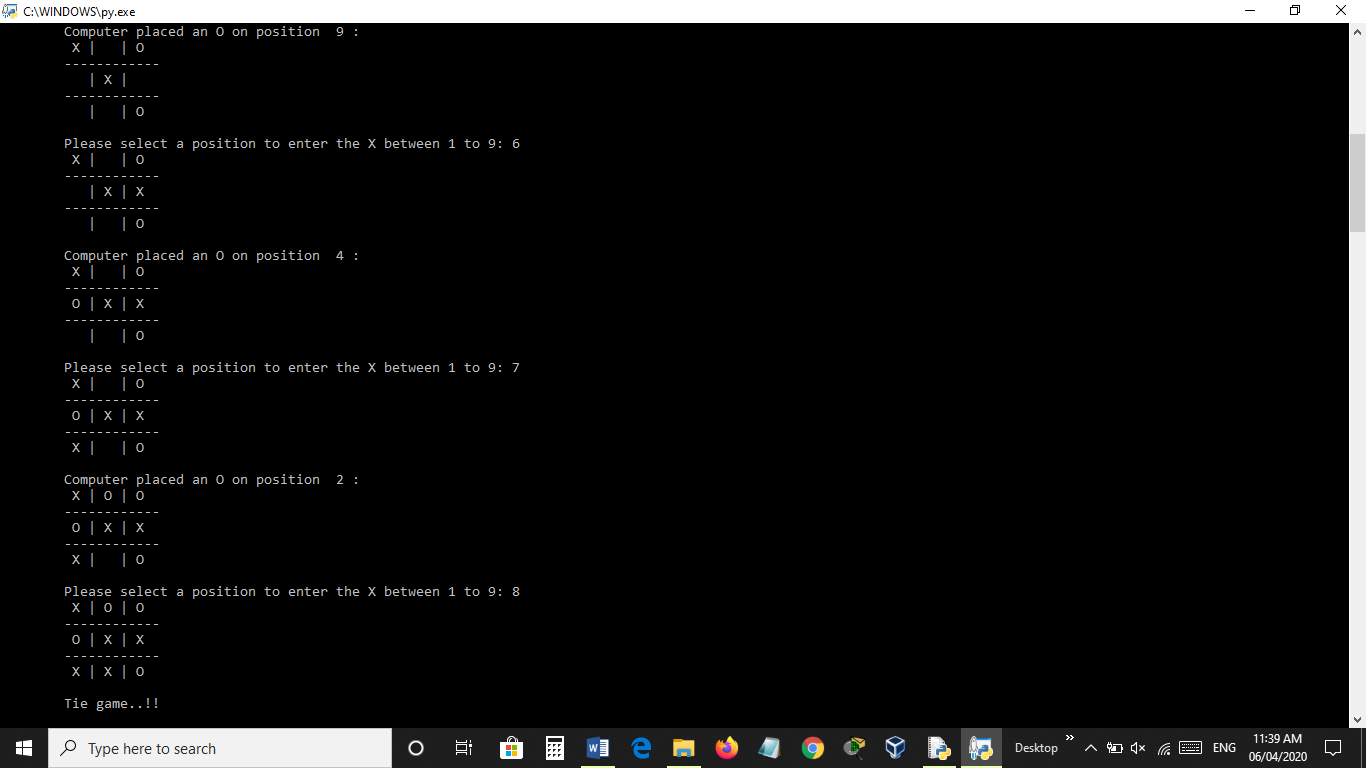


**2) WHEN PLAYER ‘X’ LOSES AND COMPUTER ‘O’ WINS -**



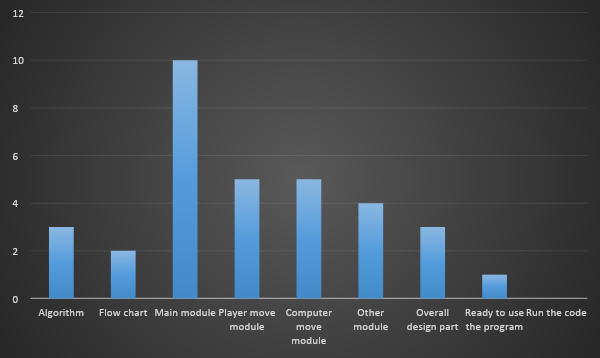


**3) WHEN PLAYER ‘X’ AND COMPUTER ‘O’ -> TIE GAME -**

**GANTT CHART**

|  |  |  |
| --- | --- | --- |
| **Start date** | **Days to complete** | **Topic** |
| 02-03-2020 | 3 days | Algorithm |
| 05-03-2020 | 2 days | Flow chart |
| 06-03-2020 | 10 days | Main module |
| 16-03-2020 | 5 days | Player move module |
| 21-03-2020 | 5 days | Computer move module |
| 26-03-2020 | 4 days | Other module |
| 30-03-2020 | 3 days | Overall design part |
| 02-04-2020 | 1 day | Ready to use the program |
| 05-04-2020 | 0 day | Run the code |

****

**Summary**

Creating a program that can play a game comes down to carefully considering all the possible situations the AI can be in and how it should respond in each of those situations. The Tic-Tac-Toe AI is simple because there are not many possible moves in Tic-Tac-Toe as compared to a game like chess or checkers.

Our AI checks if any possible move can allow itself to win. Otherwise, it checks if it must block the player’s move. Then the AI simply chooses any available corner space, then the center space, then the side spaces. This is a simple algorithm for the computer to follow.

The key to implementing our AI is by making copies of the board data and simulating moves on the copy. That way, the AI code can see if a move results in a win or loss. Then the AI can make that move on the real board. This type of simulation is effective at predicting that is it a good move or not.