**B.Tech Program First Year**
**Course: Experiential Learning**
**Course Code: DA1001**

# "UNBEATABLE TIC-TAC-TOE GAME"

by

**Sarvagya Gaur**

(Reg. No.: 209301453)

Under the guidance

of

**Mr. Satpal Singh Kushwaha**

**Assistant Professor**

Dept of CSE, SCIT, Manipal University Jaipur, Jaipur (Raj.).

**Department of Computer Science and Engineering**

**School of Computing and IT**

**Faculty of Engineering**

**Manipal University Jaipur, India**

Feb. 2021

# CERTIFICATE

This is to certify that the project titled **"Unbeatable tic-tac-toe game"** is a record of the bona fide work done by **Sarvagya Gaur (Reg No:209301453)** submitted for the partial fulfilment of the requirements for the completion of the **Experiential Learning (DA1001)** course in the Department of **Computer Science and Engineering** of Manipal University Jaipur, during the academic session July-November 2020.

*Signature of the mentor*

Satpal Singh Kushwaha
Assistant Professor
Department of CSE

*Signature of the HoD*

Dr. Sandeep Joshi
Head of the Department
Department of CSE

## ACKNOWLEDGEMENT

Working on this project has been a great learning experience for me, and for that I must express my sincere thanks to **Mr. Satpal Singh Kushwaha (Assistant Professor)** Dept of CSE, SCIT, Manipal University Jaipur, Jaipur (Raj.) as well as my teammate Rajat Yadav.

I would also like to extend my gratitude towards my family for their invaluable support.

**Sarvagya Gaur**

**(Reg. No.: 209301453)**

# TABLE OF CONTENTS

## ABSTRACT

Tic-tac-toe (also known as noughts and crosses or Xs and Os) is a paper-and-pencil game for two players, X and O, who take turns marking the spaces in a 3×3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game.

In this project, a tic tac toe game has been coded in C language. Users can play against the computer but there is a catch; The computer can never lose. This has been done to make a simple looking tic-tac-toe game a little bit more interesting and challenging.

It would be randomly decided who goes first: the player or the computer. The user would be asked the coordinates (row number and column number respectively) of the position where they want to place their ‘O’, then, the tic-tac-toe grid would be displayed on screen to show the computer’s response to the user. The game would continue in this way until it ends in either a draw or a win for the computer. All of the possible situations and outcomes in tic-tac-toe have been considered to ensure that the computer can never lose, irrespective of who goes first.

# 1. INTRODUCTION

Games played on three-in-a-row boards can be traced back to ancient Egypt, where such game boards have been found on roofing tiles dating from around 1300 BCE.

An early variation of tic-tac-toe was played in the Roman Empire, around the first century BC. It was called terni lapilli (three pebbles at a time), and instead of having any number of pieces, each player only had three, thus they had to move them around to empty spaces to keep playing. The game's grid markings have been found chalked all over Rome.

In 1952, OXO (or Noughts and Crosses), developed by British computer scientist Sandy Douglas for the EDSAC computer at the University of Cambridge, became one of the first known video games. The computer player could play perfect games of tic-tac-toe against a human opponent.

In 1975, tic-tac-toe was also used by MIT students to demonstrate the computational power of Tinkertoy elements. The Tinkertoy computer, made out of (almost) only Tinkertoys, is able to play tic-tac-toe perfectly. It is currently on display at the Museum of Science, Boston.

There are only 138 terminal board positions in the game of tic-tac-toe. A study of the game shows that when "X" makes the first move every time, the game outcomes are as follows:

- 91 distinct positions are won by (X)
- 44 distinct positions are won by (O)
- 3 distinct positions are drawn (often called a "cat's game")

In case of this project, the computer cannot lose, hence we only need to consider 94 out of the 138 outcomes, which involve either X winning or drawing.

# 2. SOFTWARE AND HARDWARE REQUIREMENTS

## 2.1 Software Requirements: -

IDE requirements - C-Free 5.0

Compiler requirements – mingw5/bcc5

OS requirements – Windows 7 or later

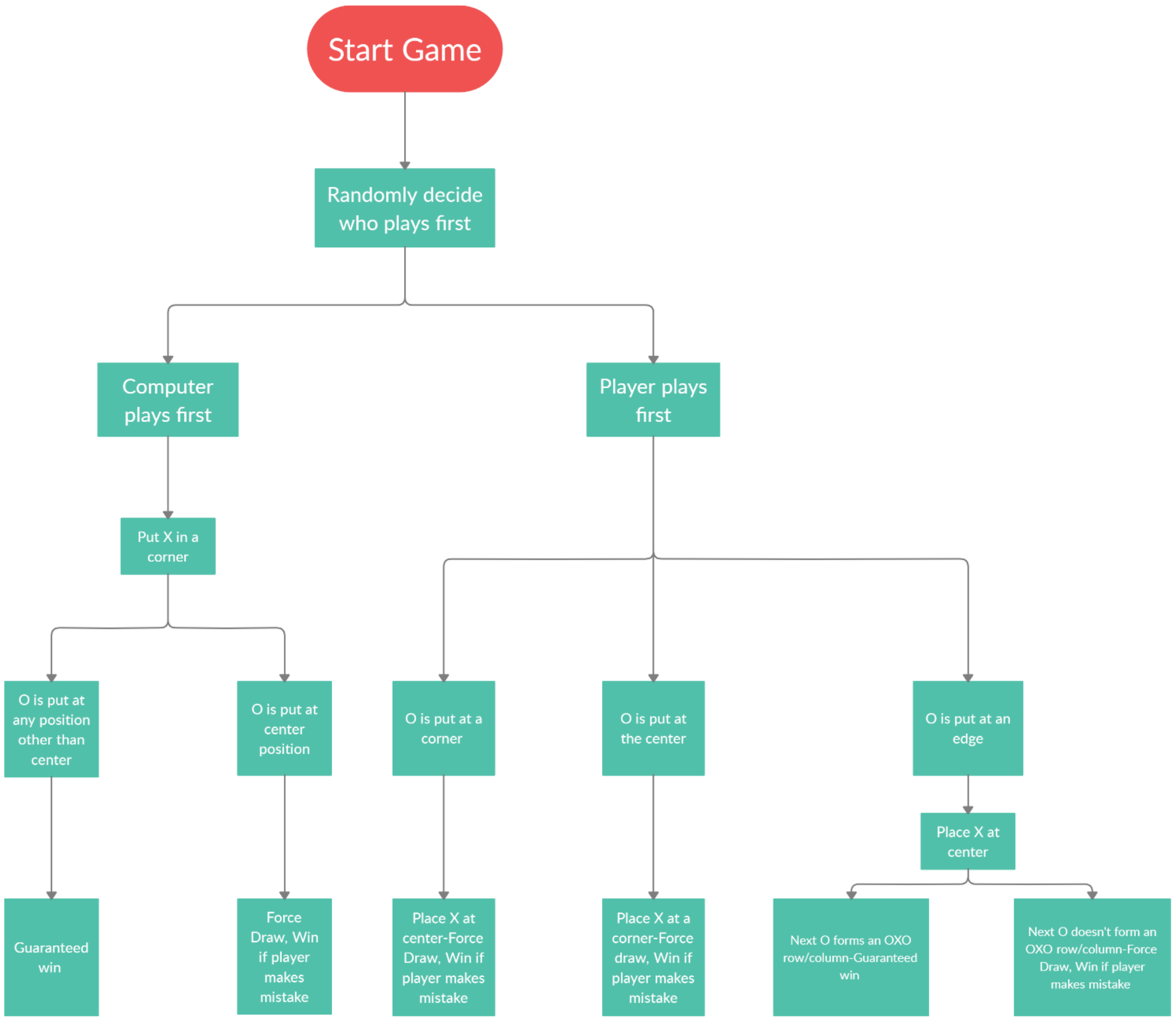
Header file requirements – stdio.h, stdlib.h, conio.h, math.h

## 2.2 Hardware Requirements: -

| Component | Recommended | Minimum |
|---|---|---|
| **CPU (no Celeron)** | Intel Core i5 or AMD Ryzen 5 | Intel Atom or AMD Ryzen 3 |
| **RAM** | 4Gb | 2Gb |
| **Hard Disk** | 32Gb or better SSD | 2Gb or better SSD |
| **Display** | 15.6" | 15.6" |
| **Video RAM** | NA | NA |

# 3. METHODOLOGY

## 3.1 Flowchart:


Start Game
Randomly decide who plays first
Computer plays first
Player plays first
Put X in a corner
O is put at any position other than center
O is put at center position
O is put at a corner
O is put at the center
O is put at an edge
Place X at center
Guaranteed win
Force Draw, Win if player makes mistake
Place X at center-Force Draw, Win if player makes mistake
Place X at a corner-Force draw, Win if player makes mistake
Next O forms an OXO row/column-Guaranteed win
Next O doesn't form an OXO row/column-Force Draw, Win if player makes mistake

## 3.2 Algorithms:

```
1.   Start
2.   Create Tic-tac-toe grid using 5*5 matrix
3.   Decide who plays first
4.   Ask user to input O at desired location
5.   Check if player would win in next turn by calling check function
6.   If yes, prevent it by putting X at the winning position
7.   Endif
8.   Check if computer can win by calling 2nd check function
9.   If yes, put X at the position and win the game
10.  Display the final matrix by calling display function
11.  Exit
12.  Else
13.  Put X at other appropriate location
14.  Endif
15.  If game not won before 9 moves
16.  Display Draw message
17.  Display final matrix by calling display function
18.  Endif
```

19. Ask if user wants to play again

20. If yes, goto step 3

21. Else

22. Stop

## 3.2 Functions:

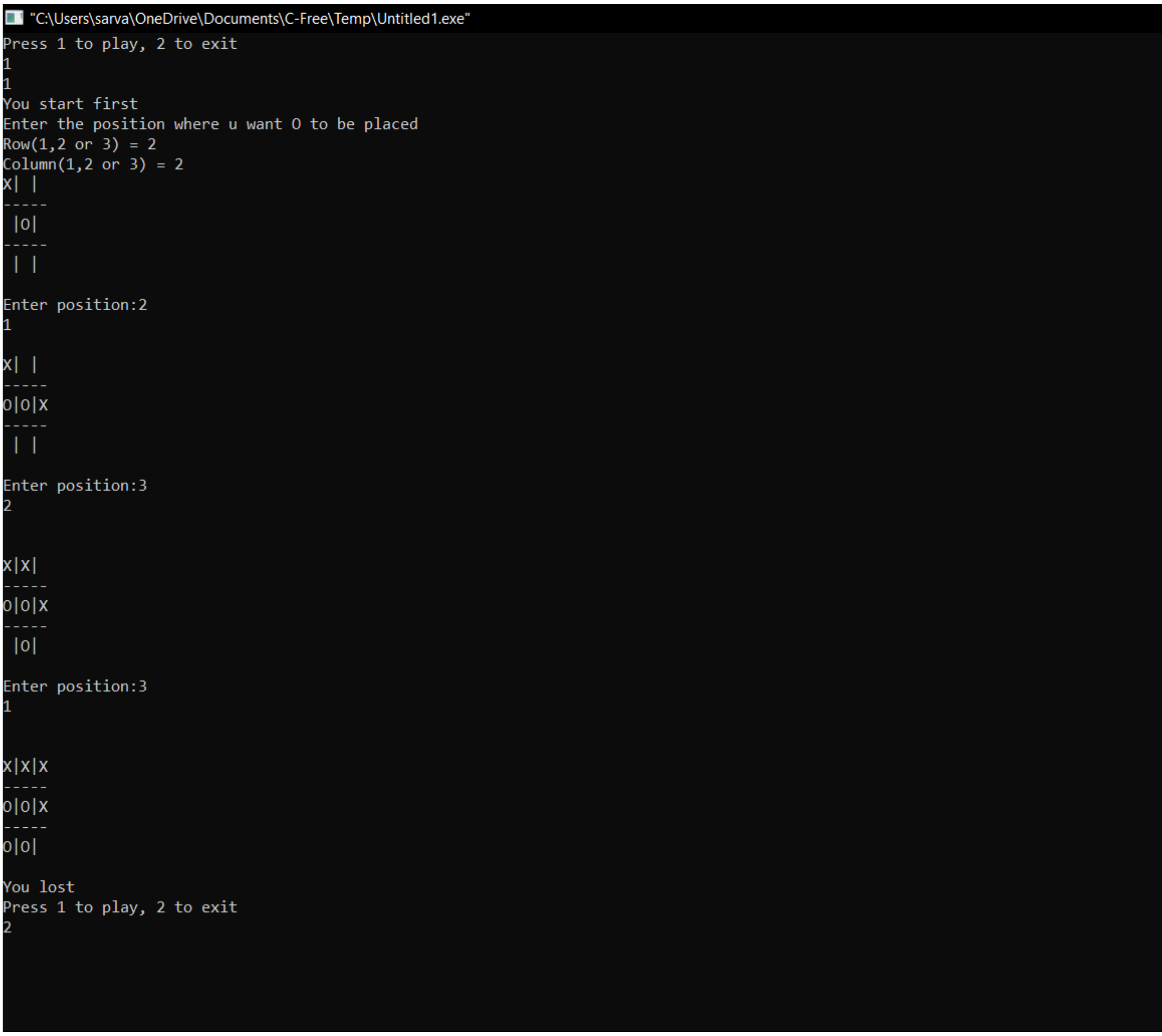
1) main() – where the game takes place, the user is asked for his/her inputs here, then the other functions are called to make the gameplay possible.

2) check() – used to check whether the player can win in the next turn or not; If yes, then a number between 1 to 9 is returned to the main function, which gives the winning position to the computer so that it can prevent the player from winning. If no, then it returns 0, which means that the player cannot win in the next turn.

3) check1() – used to check whether the computer can win in the next turn or not; If yes, then a number between 1 to 9 is returned to the main function, which gives the winning position to the computer so that it can mark it's X at that position and win. If no, then it returns 0, which means that the computer cannot win in the next turn.

4) display() – After the end of each of turn, the player should be able to see his/her move as well as the computer's move reflected on the tic tac toe grid. This is what the display() function is used for, as it is called to display the current progress of the tic-tac-toe grid.

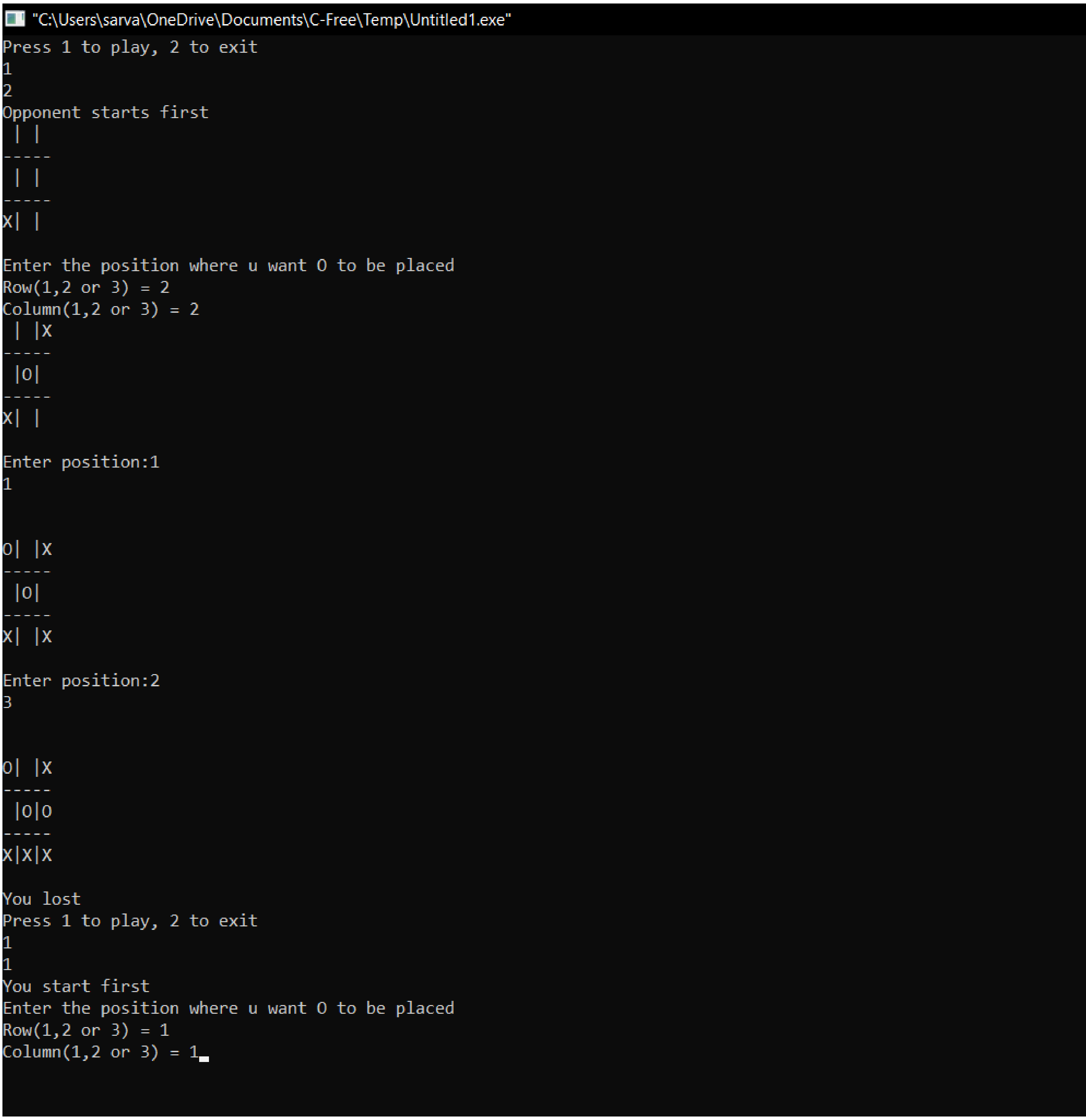
# 4. RESULTS AND DISCUSSIONS



"C:\Users\sarva\OneDrive\Documents\C-Free\Temp\Untitled1.exe"

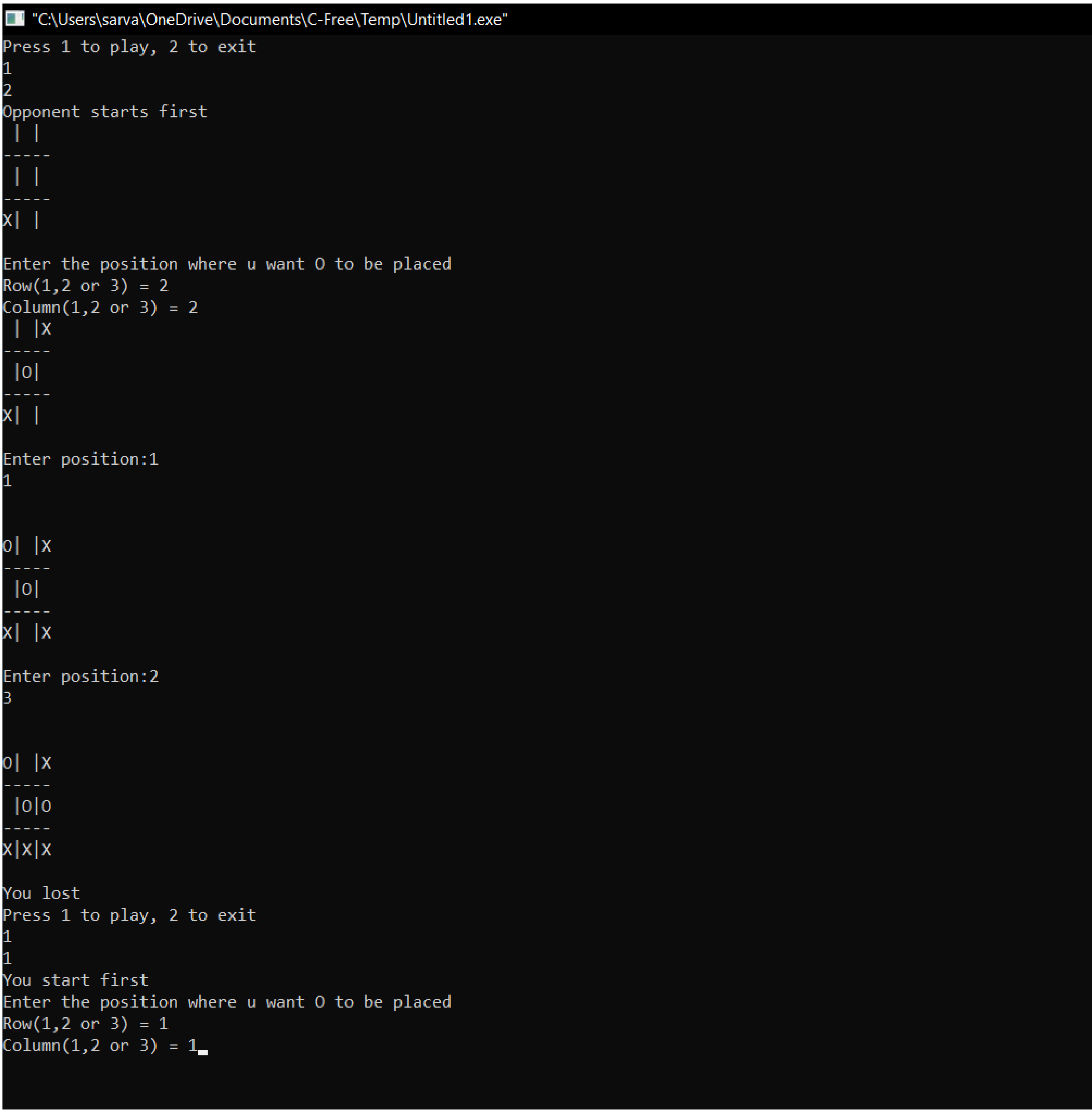
```
Press 1 to play, 2 to exit
1
1
You start first
Enter the position where u want O to be placed
Row(1,2 or 3) = 2
Column(1,2 or 3) = 2
X| |
-----
 |O|
-----
 | |

Enter position:2
1

X| |
-----
O|O|X
-----
 | |

Enter position:3
2


X|X|
-----
O|O|X
-----
 |O|

Enter position:3
1


X|X|X
-----
O|O|X
-----
O|O|

You lost
Press 1 to play, 2 to exit
2
```

-



"C:\Users\sarva\OneDrive\Documents\C-Free\Temp\Untitled1.exe"

```
Press 1 to play, 2 to exit
1
2
Opponent starts first
 | |
-----
 | |
-----
X| |

Enter the position where u want O to be placed
Row(1,2 or 3) = 2
Column(1,2 or 3) = 2
 | |X
-----
 |O|
-----
X| |

Enter position:1
1


O| |X
-----
 |O|
-----
X| |X

Enter position:2
3


O| |X
-----
 |O|O
-----
X|X|X

You lost
Press 1 to play, 2 to exit
1
1
You start first
Enter the position where u want O to be placed
Row(1,2 or 3) = 1
Column(1,2 or 3) = 1
```



```
"C:\Users\sarva\OneDrive\Documents\C-Free\Temp\Untitled1.exe"
Press 1 to play, 2 to exit
1
2
Opponent starts first
 | |
-----
 | |
-----
X| |

Enter the position where u want 0 to be placed
Row(1,2 or 3) = 2
Column(1,2 or 3) = 2
 | |X
-----
 |0|
-----
X| |

Enter position:1
1


0| |X
-----
 |0|
-----
X| |X

Enter position:2
3


0| |X
-----
 |0|0
-----
X|X|X

You lost
Press 1 to play, 2 to exit
1
1
You start first
Enter the position where u want 0 to be placed
Row(1,2 or 3) = 1
Column(1,2 or 3) = 1
```

## 5. CONCLUSION

The Tic-Tac-Toe game is most familiar among all the age groups. An algorithm of playing Tic-Tac-Toe has been presented and tested that works in efficient way. Overall the system works without any bugs. The game is developed as a challenge and subsequently for entertainment. It teaches the Gamer to be alert at every situation he/she faces, because if the Gamer is not fully alert then they will instantly lose. The game does not make use of any graphics or sound engines, the tic tac toe grid is in fact made using the pipe bar and underscore characters. Hence, the system requirements required to play it are very basic. Kids can also play this game because the design of the game is very simple. As the user is required to enter coordinates of the position where they want to mark their O’s, this game can work as a fun and efficient method to introduce kids to the concept of matrices.

## 6. FUTURE PROSPECTS

Around 2600 lines of code were required to make this project possible. Although 3 functions were defined outside the main function which were repeatedly called to shorten the code, still it featured a bulk of iterations; Much of the content of the code was copied and pasted over and over again. As a result, the code is tedious to read and difficult to comprehend.

A solution to the above problems would be to use what's called a minimax algorithm. This algorithm is based on the concept of recursion. In programming languages, if a program allows you to call a function inside the same function, then it is called a recursive call of the function. This can be very useful in solving many mathematical problems such as factorial, Fibonacci series, etc.

The Minimax algorithm sees a few steps ahead and puts itself in the shoes of its opponent. It keeps playing ahead until it reaches a terminal arrangement of the board resulting in a tie, a win, or a loss. Once in a terminal state, the AI will assign an arbitrary positive score (+10) for a win, a negative score (-10) for a loss, or a neutral score (0) for a tie. At the same time, the algorithm evaluates the moves that lead to a terminal state based on the players' turn. It will choose the move with maximum score when it is the AI's turn and choose the move with the minimum score when it is the human player's turn. Using this strategy, Minimax avoids losing to the human player.

Therefore, the minimax algorithm would achieve the same goal, but do it using less lines of code and make it more comprehensible.

## 7. REFERENCES

1. www.google.com
2. C Programming Language - GeeksforGeeks
3. https://stackoverflow.com/
4. https://www.wikihow.com/Win-at-Tic-Tac-Toe
5. https://www.tutorialspoint.com/cprogramming/c_recursion.htm
6. https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm-9d690bad4b37/
7. https://creately.com/diagram-type/flowchart