A

Synopsis/Project Report

On

TIC TAC TOE

Submitted in partial fulfillment of the requirement for the IVth semester

Bachelor of Computer Science

By

Neeraj Koshyari

Priyanshu

Under the Guidance of

Mr. Devesh Pandey

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2022- 2023

STUDENT'S DECLARATION

We, **Neeraj koshyari and Priyanshu** here-by declare the work, which is being presented in the project, entitled "**TIC TAC TOE**" in partial fulfillment of the requirement for the award of the degree **B.Tech** in the session **2022-2023**, is an authentic record of our own work carried out under the supervision of "**Mr. Devesh Pandey", Assistant Professor, Department of CSE, Graphic Era Hill University, Bhimtal.**

The matter embodied in this project has not been submitted by us for the award of any other degree.

Date: …… …….

(Full signature of student)

CERTIFICATE

The project report entitled "TIC TAC TOE" being submitted by Neeraj Koshyari and Priyanshu to Graphic Era Hill University Bhimtal Campus for the award of bonafide work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of report.

(Mr. Devesh Pandey) **(Dr. Ankur Bisht)**

Project Guide **(HOD, CSE Dept.)**

ACKNOWLEDGEMENT

We take immense pleasure in thanking Honorable **"Mr. Devesh Pandey"** (**AssistantProfessor,CSE, GEHU Bhimtal Campus**) to permit me and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me everything that we need. We again want to extend thanks to our President **"Prof. (Dr.) Kamal Ghanshala"** for providing us all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to Professor **"Dr. Manoj Chandra Lohani"** (Director Gehu Bhimtal), other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

Finally, yet importantly, we would like to express my heartiest thanks to our beloved parents, for their moral support, affection and blessings. We would also like to pay our sincere thanks to all our friends and well-wishers for their help and wishes for the successful completion of this research.

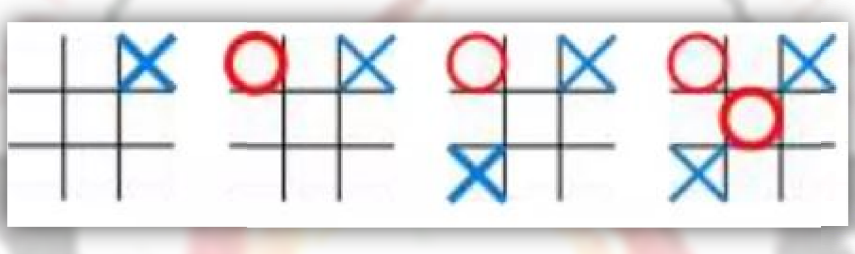
Neeraj Koshyari

Priyanshu

ABSTRACT

This report is an introduction to the Tic-Tac-Toe game in C++ programming. Anybody, who doesn't know even the basics of Tic Tac Toe in c++ will certainly be able to understand and gain great knowledge from this report. **Tic Tac Toe** (also known as **Naught and Crosses** or **X and O**) is a paper and pencil game for two players, X and O, who takes turns marking the spaces in a 3 x 3 grid. The player who succeeds in placing three of his marks in vertical, horizontal or diagonal wins the game.

The following example of the game won by the X player:



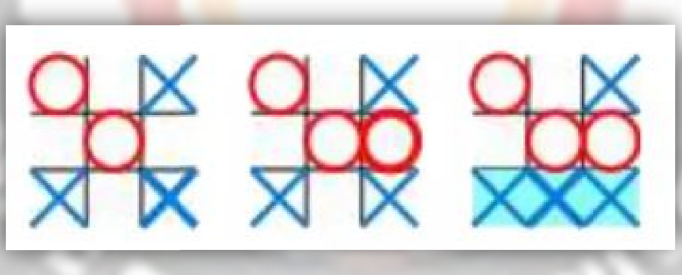


Table of CONTENTS

INTRODUCTION

Because of the simplicity of tic-tac-toe, it is often used as pedagogical tool for teaching the concepts of good sportsmanship and the branch of artificial intelligence that deals with the searching of game trees. It is straightforward to write a computer program to play tic-tac-toe perfectly, to enumerate the 765 essentially different positions (the state space complexity) or the 26,830 possible games up to rotations and reflections (the game tree complexity) on this space.

The game can be generalized to an m,n,k-game in which two players alternate placing stones of their own color on an m x n board. With the goal getting k of their own color in a row. Tic-tac-toe is the (3,3,3)-game. Harary's generalized tic-tac-toe is an even broader generalization of tic tac toe. It can also be generalized as n^d game. Tic-tac-toe is the game where n equals 3 and d equals 2. If played properly the game will end in draw making tic-tac-toe a futile game.

WHAT IS A TIC TAC TOE GAME?

Tic-tac-toe is not a very challenging game for human beings. If you're an enthusiast, you've probably moved from the basic game to some variant like three dimensional tic-tac-toe on a larger gird. If you sit down right now to play ordinary three-by-three tic-tac-toe with a friend, what will probably happen is that every game will come out a tie.

Both you and your friend can probably play perfectly, never making a mistake that would allow your opponent to win. But can describe how you know here to move each turn? Most of the time, you probably aren't even aware of alternative possibilities; you just look at the board and instantly know where you want to move. That kind of instant knowledge is great for human beings, because it makes you a fast player. But it isn't much help in writing a computer program. For that, you have to know very explicitly what your strategy is.

TOOLS/ PLATFORM, HARDWARE AND SOFTWARE REQUIREMENT SPECIFICATIONS

Tools

- Android Studio(software)
- Android Phone

Platform

- Windows 7/8/10/11

Hardware Requirement Specification

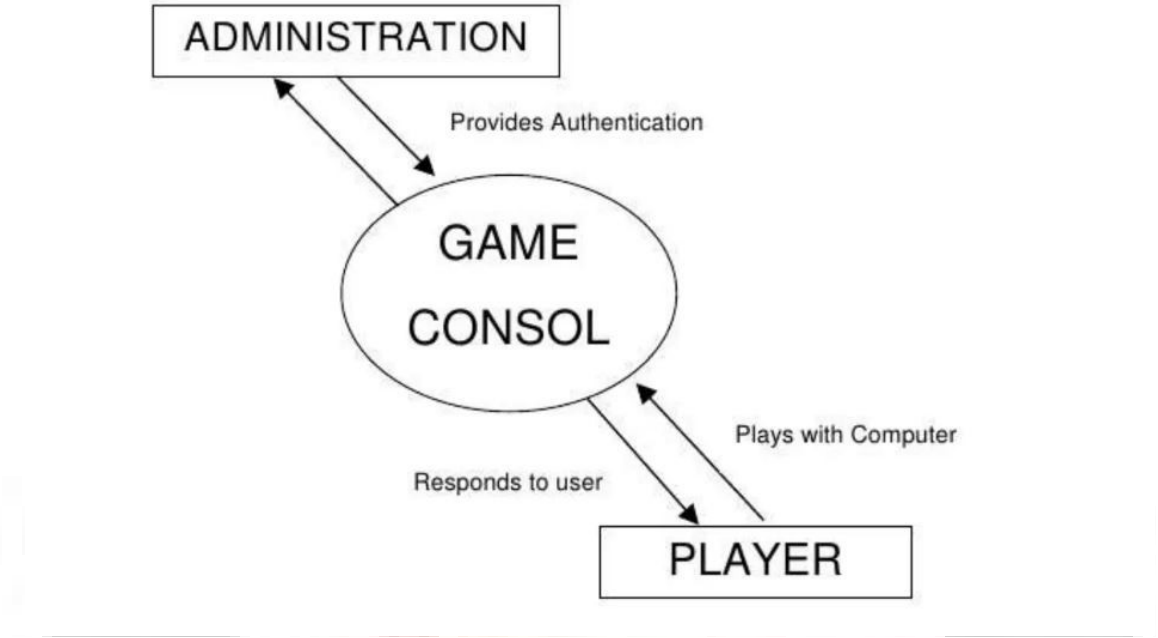
Criterion	Description
Disk Space	500 MB disk space for Android Studio, at least 1.5 GB for Android SDK, emulator system images, and caches
RAM	3GB RAM minimum, 8 GB RAM recommended, plus 1GB for the Android Emulator
CPP Version	C++23

Software Requirement Specification

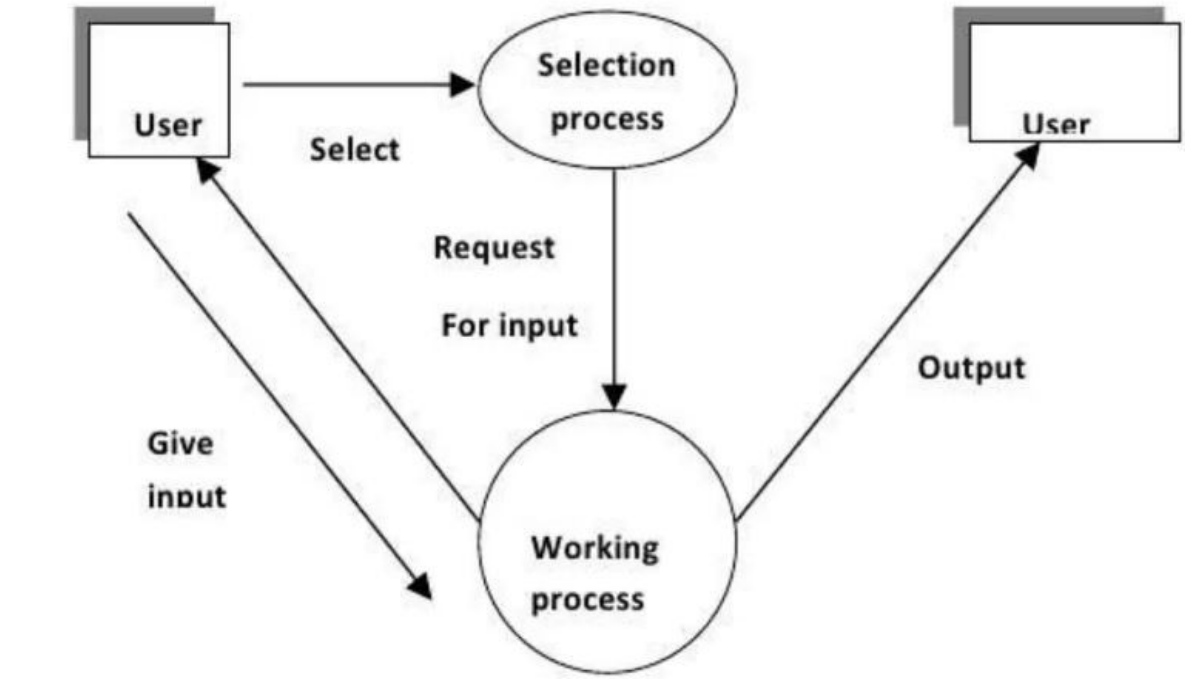
Latest version of cpp.

Data Flow Diagram

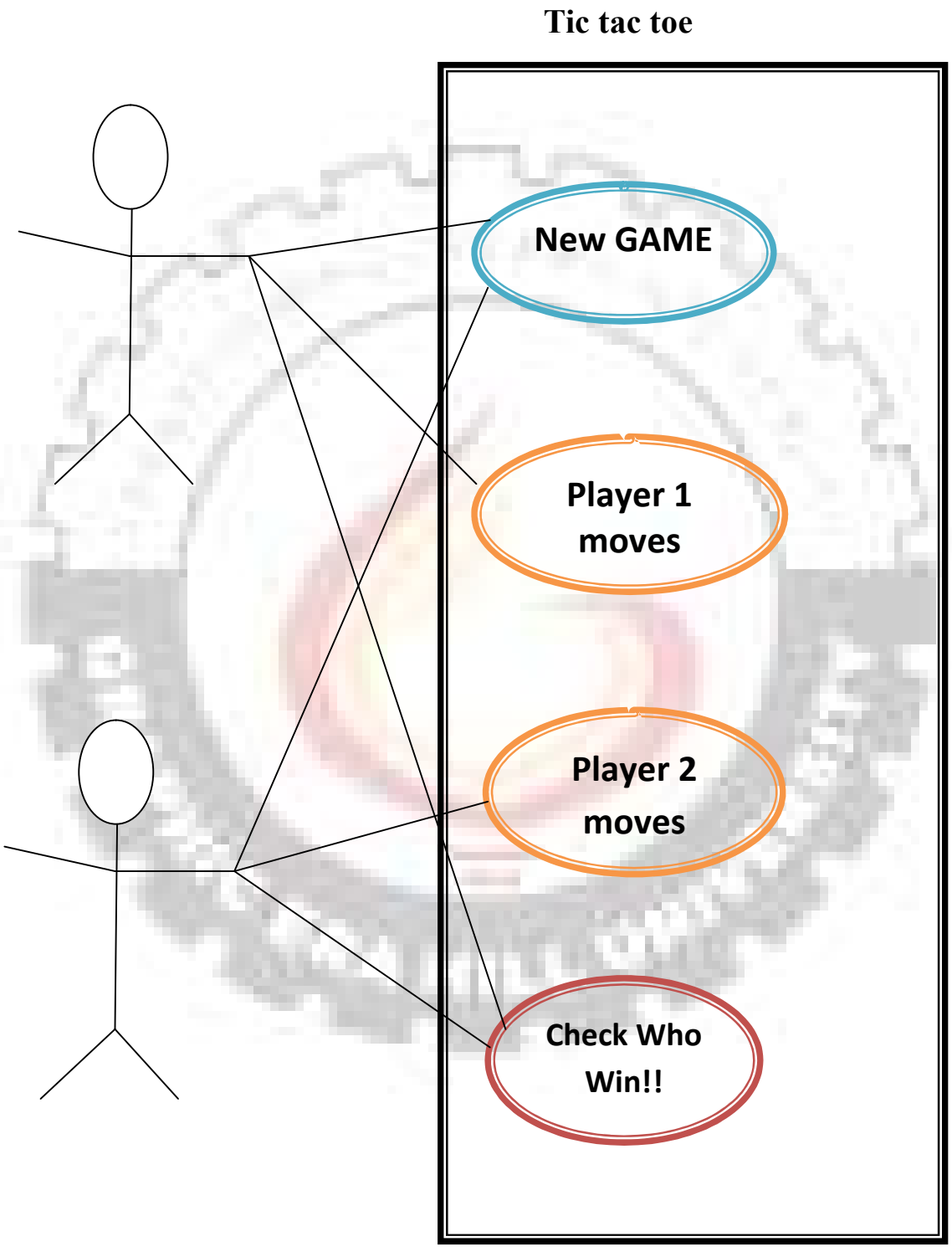
0 Levels DFD:-



1 Levels DFD:-



CASE DIAGRAM



ALGORITHM USED IN TIC TAC TOE

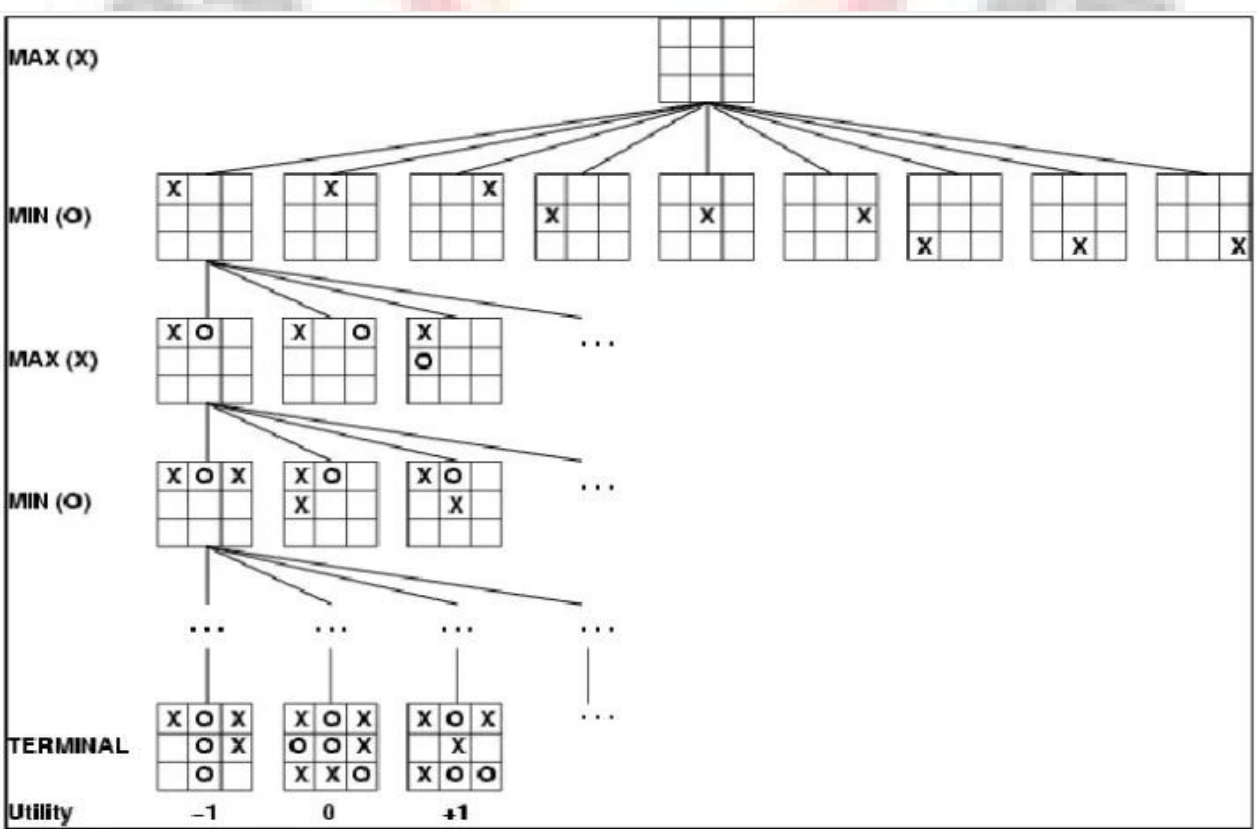
Algorithm used here is minimax algorithm to compute the optimal move of a player and decide the winner.

Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. It is widely used in two player turn based games such as TIC-TAC_TOE, Backgamon, Mancala, Chess, etc.

In Minimax the two players are called maximize and minimize. The maximize tries to get the higest score possible while the minimize tries to get the lowest score possible wile minimize to do opposite.

Every Board state has a value associated with it. In a given state if the maximize has upper hand than, the score of the board will tend to be some positive value. If the minimize has the upper hand in that board state then it will tend to be some negative value. The values of the board are calculated by some heuristics which are unique for every type of game.

GAME TREE



MINIMAX TERMINOLOGY

- Utility function: the function applied to leaf nodes
- Backed up values
- Of a max position- the value of its largest successor
- Of a min position- the value of its smallest successor.

MINIMAX PROCEDURE

- Search down several levels.
- Apply utility function to leaf nodes.
- Back up values all the way up to the root node.
- That node selects the move.

PROPERTIES OF MINIMAX

It is complete algorithm if tree is finite. It is optimal if you are playing with an optimalplayer.Time complexity computed here is O(bm). Space complexity computed here is O(dm).Where b is the number of legal moves available for a player at each point and d is themaximum depth of the tree.

ADVANTAGE

The Minimax algorithm helps find the best move, by working backwards from the endof the game. At each step it assumes that player A is trying to maximize the chancesof its winning, while on the next turn player B is trying to minimize the chances of A's winning.

DISADVANTAGE

A disadvantage of the minimax algorithm is that each board state has to be visitedtwice: one time to find its children and a second time to evaluate the heuristic value.

CODE

```
#include<bits/stdc++.h>
using namespace std;


vector<vector<char>> board;
int choice;
int row,column;
char turn;
bool draw = false;
string s1,s2;


void display(){
   cout<<"----------\t----------\n\n";
   cout<<"\t\t    |    |    \n";
   cout<<"\t\t  "<<board[0][0]<<"  |  "<<board[0][1]<<"  |  "<<board[0][2]<<"\n";
   cout<<"\t\t_____|_____|_____\n";
   cout<<"\t\t    |    |    \n";
   cout<<"\t\t  "<<board[1][0]<<"  |  "<<board[1][1]<<"  |  "<<board[1][2]<<"\n";
   cout<<"\t\t_____|_____|_____\n";
   cout<<"\t\t    |    |    \n";
   cout<<"\t\t  "<<board[2][0]<<"  |  "<<board[2][1]<<"  |  "<<board[2][2]<<"\n";
   cout<<"\t\t    |    |    \n";
}


void instruction(){
   cout<<"----------\t----------\n\n";
   cout<<"\t\t     |     |     \n";
```

```
    cout<<"\t\t  00  |  01  |  02  \n";
    cout<<"\t\t______|______|______\n";
    cout<<"\t\t      |      |      \n";
    cout<<"\t\t  10  |  11  |  12  \n";
    cout<<"\t\t______|______|______\n";
    cout<<"\t\t      |      |      \n";
    cout<<"\t\t  20  |  21  |  22  \n";
    cout<<"\t\t      |      |      \n";
}

void player_turn(){
    if(turn == 'X')
        cout<<"\nPlayer of current turn: "<<s1<<"\n";
    else if(turn == 'O')
        cout<<"\nPlayer of current turn: "<<s2<<"\n";


    string row1,column1;
    while(1){
        cout<<"\nChoose a row number (0 to 2):\n";
        getline(cin,row1);
        if(row1[0]<'0' || row1[0]>'2' || row1.size()!=1){
            cout<<row1<<" is not a valid row.\n";
            continue;
        }

        cout<<"Choose a column number (0 to 2):\n";
```

```
        getline(cin,column1);
        if(column1[0]<'0' || column1[0]>'2' || column1.size()!=1){
            cout<<column1<<" is not a valid column.\n";
            continue;
        }
        break;
    }
    int row=row1[0]-'0';
    int column=column1[0]-'0';

    if(turn == 'X' && board[row][column] != 'X' && board[row][column] != 'O'){
        board[row][column] = 'X';
        turn = 'O';
    }else if(turn == 'O' && board[row][column] != 'X' && board[row][column] != 'O'){
        board[row][column] = 'O';
        turn = 'X';
    }else {
        cout<<"Box already filled!\nPlease choose another!!\n\n";
        player_turn();
    }

    display();
}

bool empty(){
    //for simple row and column
    for(int i=0; i<3; i++){
```

```
        int val1=(board[i][0] + board[i][1] + board[i][2]);//row
        int val2=(board[0][i] + board[1][i] + board[2][i]);//column
        if(val1== 264 || val2==264 || val1==237 || val2==237)
            return false;
    }

    //for both diagoonals
    int val1=(board[0][0] + board[1][1] + board[2][2]);
    int val2=(board[0][2] + board[1][1] +  board[2][0]);
    if( val1==264 || val2==264 || val1==237 || val2==237 )
            return false;

    //if any space empty
    for(int i=0; i<3; i++){
            for(int j=0; j<3; j++){
                    if(board[i][j] != 'X' && board[i][j] != 'O')
                            return true;
                }
        }

    draw = true;
    return false;
}


bool YorN(){
    char ch='c';
    while(ch!='Y' || ch!='N'){
```

```
        cout<<"\n\nWould you like to play again? (Y/N)"<<endl;
        cin>>ch;
        if(ch=='Y')
            return true;
        else if(ch=='N')
            return false;
        else
            cout<<ch<<" is not a valid answer\n";
    }
    return true;
}

int main()
{
    cout<<"------------------------Instruction---------------------------\n\n";
    cout<<"Every Box is represented by a row and a column respectively as shown below!!\n";
    cout<<"Select the row number and column number accordingly!!\n";
    instruction();
    while(1){
        string s;
        s1="",s2="";
        while(s1==s2){
            cout<<"\nEnter the name of 'X' player:\n";
            getline(cin,s1);
            if(!(s1[0]>='a' && s1[0]<='z') && !(s1[0]>='A' && s1[0]<='Z')){
                cout<<"Frist character of name should start with an Alphabet!!\n";
                s1="";
```

```
        continue;
    }
    cout<<"\nEnter the name of 'O' player:\n";
    getline(cin,s2);
    if(!(s2[0]>='a' && s2[0]<='z') && !(s2[0]>='A' && s2[0]<='Z')){
        cout<<"Frist character of name should start with an Alphabet!!\n";
        s1=s2;
    }
    else if(s1==s2)
        cout<<"Both should have different name!!\n";
}

while(s!=s1 && s!=s2){
    cout<<"\nWho plays first, "<<s1<<" or "<<s2<<"?\n";
    getline(cin,s);
    if(s!=s1 && s!=s2)
        cout<<s<<" is not a registered player.\n";
}
if(s==s1)
    turn='X';
else    turn='O';

board={{' ',' ',' '},{' ',' ',' '},{' ',' ',' '}};
cout<<"\n"<<"    "<<s1<<": [X]  "<<s2<<": [O]\n\n";
cout<<"Game board\n";
display();
while(empty()){
```

```
        player_turn();
        empty();
    }
    cout<<"\n\n\tGame is over:\n";
    if(turn == 'O' && draw == false)
        cout<<"\t"<<s1<<" wins!";
    else if(turn == 'X' && draw == false)
        cout<<"\t"<<s2<<" wins!";
    else
        cout<<"\t"<<"it is a tie!";
    if(YorN())  continue;
    else    break;
  }
  cout<<"Bye!\n";
  return 0;
}
```

TESTING

TEAM INTERATION

The following describes the level of team interaction necessary to have a successful product.

- The test team will work closely with the Development team to acviece a high quality design and user interface specifications based on customer requirements. The test team is responsible for visualizing test cases and raising quality issues and concerns during meetings to address issues early enough in the development cycle.

- The Test team will work closely with Development Team to determine whether or not the application meets standards for completeness. If an area is not acceptable for testing, the code complete date will be pushed out, giving the developers additional time to stabilize the area.

- Since the application with a back-end system component, the Test Team will need to include a plan for integration testing. Integration testing must be executed successfully prior to system testing.

TESTING OBJECTIVE

The objective our test plan is to find and report as many bugs as possible to improve the integrity of our program.

ENHENCEMENTS

- Colors can be added in rows and column.
- Background music can be while playing the game.
- GUI bases
- Touch game etc.

CONCLUSION

We have successfully implemented the Tic-Tac-Toe game in C++ language.

REFERENCES

- Google.com
- Wikipedia.org
- Stackoverflow.com
- Geeks for Geeks