**CHAPTER-4**

**4. SOURCE CODE**

#include <stdio.h>

#include <conio.h>

#include <windows.h>

#include <process.h>

char square[10] = { 'o', '1', '2', '3', '4', '5', '6', '7', '8', '9' };

int checkwin();

void board();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

THIS FUNCTION IS FOR THE LOADING PAGE

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void gotoxy(int x,int y)

{

COORD coord;

coord.X=x;

coord.Y=y;

SetConsoleCursorPosition(GetStdHandle(STD\_OUTPUT\_HANDLE),coord);

}

int main()

{ int r,q;

gotoxy(50,10);

printf("LOADING-----");

gotoxy(50,11);

(11)

for(r=1;r<=20;r++)

{ for(q=0;q<=100000000;q++);

printf("%c",177);

}

int player = 1, i, choice;

char mark;

do

{

board();

player = (player % 2) ? 1 : 2;

printf("Player %d, Enter your move number: ", player);

scanf("%d", &choice);

mark = (player == 1) ? 'X' : 'O';

if (choice == 1 && square[1] == '1')

square[1] = mark;

else if (choice == 2 && square[2] == '2')

square[2] = mark;

else if (choice == 3 && square[3] == '3')

square[3] = mark;

else if (choice == 4 && square[4] == '4')

square[4] = mark;

(12)

else if (choice == 5 && square[5] == '5')

square[5] = mark;

else if (choice == 6 && square[6] == '6')

square[6] = mark;

else if (choice == 7 && square[7] == '7')

square[7] = mark;

else if (choice == 8 && square[8] == '8')

square[8] = mark;

else if (choice == 9 && square[9] == '9')

square[9] = mark;

else

{

printf("Invalid move ");

player--;

getch();

}

i = checkwin();

player++;

}while (i == - 1);

board();

if (i == 1)

{

(13)

printf("==>\a\tPlayer %d win ", --player);

printf("\n\n\t!!!! Congratulations Player For The Win!!!!");}

else

{

printf("==>\a\tGame draw");

printf("\n\n\t!!!!!Better Luck Next Time Players!!!!!");}

getch();

return 0;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION TO RETURN GAME STATUS

1 FOR GAME IS OVER WITH RESULT

-1 FOR GAME IS IN PROGRESS

O GAME IS OVER AND NO RESULT

\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int checkwin()

{

if (square[1] == square[2] && square[2] == square[3])

return 1;

else if (square[1] == square[4] && square[4] == square[7])

return 1;

else if (square[7] == square[8] && square[8] == square[9])

return 1;

(14)

else if (square[3] == square[6] && square[6] == square[9])

return 1;

else if (square[2] == square[5] && square[5] == square[8])

return 1;

else if (square[4] == square[5] && square[5] == square[6])

return 1;

else if (square[1] == square[5] && square[5] == square[9])

return 1;

else if (square[3] == square[5] && square[5] == square[7])

return 1;

else if (square[1] != '1' && square[2] != '2' && square[3] != '3' &&

square[4] != '4' && square[5] != '5' && square[6] != '6' && square[7]

!= '7' && square[8] != '8' && square[9] != '9')

return 0;

else

return - 1;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTION TO DRAW BOARD OF TIC TAC TOE WITH PLAYERS MARK

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void board()

{ system("cls");

(15)

printf("!!!WELCOME PLAYERS!!! \n\n");

printf("!!!!TO THE GAME OF TIC TAC TOE!!!!\n\n");

printf("Player 1 (X) - Player 2 (O)\n\n\n");

printf(" | | \n");

printf(" %c | %c | %c \n", square[1], square[2], square[3]);

printf("\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\n");

printf(" | | \n");

printf(" %c | %c | %c \n", square[4], square[5], square[6]);

printf("\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_\n");

printf(" | | \n");

printf(" %c | %c | %c \n", square[7], square[8], square[9]);

printf(" | | \n\n");

}

/\*\*\*\*\*\*END OF PROJECT\*\*\*\*\*\*\*\*/

(16)

**CHAPTER -3**

INTRODUCTION TO TIC TAC TOE

Tic tac toe, whose name has several variations according to its player's geographical location, is a simple game game that all children have once given a try. It's name, as the topic of this article asks, has a fairly straightforward etymology, yet a long history possibly dating back to the Roman empire.

**3.1. Features**

Tic tac toe is a game played by two players, identified respectively as "X" and "O." The playing board is a 3x3 grid drawn with pencil on paper. The players alternate marking X and O within the nine squares of the grid, each intending to create a row of three marks either horizontally, vertically or diagonally. X usually goes first..

**3.2. Identification**

"Tic tac toe" has a few other accepted spellings: tick tack toe, tick tat toe and tit tat toe among other. Hyphenation of the word is also acceptable (tic-tac-toe). In England the game is commonly called "noughts and crosses." Depending on the region in Ireland, "X's and O's," "X-e O-zees" and "Boxin' Oxen" are also used. Norway calls it "Twiddles and Bears."

**3.3 History**

The name tic tac toe comes from a game by the same name, no longer played, in which players with their eyes closed tossed a pencil down onto a slate marked with numbers, and earned the score the number indicated -- something like blind darts.. "Ticktack," according to the Random House Dictionary of English language, is a repetitive sound made by repetitive tapping, knocking or clicking. Thus, "tic tac toe" is an imitation of the sound the pencil makes when hitting the slate.

**3.4. Size**

Because of its mathematical simplicity, it is simple to create a computer game that simulates tic tac toe perfectly. Examining the probability of possible games, it is possible to place X's and O's on the grid (in winning and non-winning combinations) in **362,800 unique configurations**. There are **255,168 possible winning configurations**. There are, however, only 138 unique winning outcomes if symmetrical plays are eliminated from the equation.

**3.5. Potential**

Other slightly more complex variations exist that involve tic tac toe's main objective: being the first player to form a row of so many marks. Connect Four is a popular one, as well as **Three Men's Morris, Nine Men's Morris, Pente, Qubic and Quarto.**

(10)

**CHAPTER -1**

INTRODUCTION TO C

[C](https://www.geeksforgeeks.org/c-programming-language/) is a procedural programming language. It was initially developed by Dennis Ritchie between 1969 and 1973. It was mainly developed as a system programming language to write an operating system. The main features of C language include low-level access to memory, a simple set of keywords, and clean style, these features make C language suitable for system programmings like an operating system or compiler development.  
Many later languages have borrowed syntax/features directly or indirectly from C language. Like syntax of Java, PHP, JavaScript and many other languages are mainly based on C language. C++ is nearly a superset of C language (There are few programs that may compile in C, but not in C++).

* 1. **Component of C programming:**

The components of the C programming structure are:

1. **Header Files Inclusion**: The first and foremost component is the inclusion of the Header files in a C program.  
   A header file is a file with extension .h which contains C function declarations and macro definitions to be shared between several source files.

Some of C Header files:

* + **stddef.h** – Defines several useful types and macros.
  + **stdio.h** – Defines core input and output functions
  + **stdlib.h** – Defines numeric conversion functions, pseudo-random network generator, memory allocation
  + **string.h** – Defines string handling functions
  + **math.h** – Defines common mathematical functions

**Syntax to include a header file in C:**

#include <(header\_file\_name).h>

1. **Main Method Declaration:** The next part of a C program is to declare the main() function.
2. **Variable Declaration:** The next part of any C program is the variable declaration. It refers to the variables that are to be used in the function. Please note that in C .

(05)

1. program, no variable can be used without being declared. Also in a C program, the variables are to be declared before any operation in the function.

**Example:**

int main()

{

**int a;**

.

.

1. **Body:** Body of a function in C program, refers to the operations that are performed in the functions. It can be anything like manipulations, searching, sorting, printing, etc.

**Example:**

int main()

{

int a;

**printf("%d", a);**

.

.

1. **Return Statement:** The last part in any C program is the return statement. The return statement refers to the returning of the values from a function. This return statement and return value depend upon the return type of the function. In any other case, there will be a return statement and the return value will be of the type of the specified return type.

**Example:**

int main()

{

int a;

printf("%d", a);

**return 0;**

**1.2. Arrays in C**

In C language, arrays are reffered to as structured data types. An array is defined as **finite ordered collection of homogenous** data, stored in contiguous memory locations.

(06)

Here the words,

* **finite** *means* data range must be defined.
* **ordered** *means* data must be stored in continuous memory addresses.
* **homogenous** *means* data must be of similar data type.

### Example where arrays are used,

* to store list of Employee or Student names,
* to store marks of students,
* or to store list of numbers or characters etc.

Since arrays provide an easy way to represent data, it is classified amongst the data structures in C. Other data structures in c are **structure**, **lists**, **queues**, **trees** etc. Array can be used to represent not only simple list of data but also table of data in two or three dimensions.

### Declaring an Array

Like any other variable, arrays must be declared before they are used. General form of array declaration is,

data-type variable-name[size];

/\* Example of array declaration \*/

int arr[10];

### Initialization of an Array

After an array is declared it must be initialized. Otherwise, it will contain **garbage** value(any random value). An array can be initialized at either **compile time** or at **runtime**.

#### Compile time Array initialization

Compile time initialization of array elements is same as ordinary variable initialization. The general form of initialization of array is,

data-type array-name[size] = { list of values };

/\* Here are a few examples \*/

int marks[4]={ 67, 87, 56, 77 }; // integer array initialization

float area[5]={ 23.4, 6.8, 5.5 }; // float array initialization

int marks[4]={ 67, 87, 56, 77, 59 }; // Compile time error

### (07)

### 

# **. The if Function**

The if keyword in the C programming language is used to make decisions in your code based upon simple comparisons. It’s the same concept humans use in making decisions based on the question “what if?”

Here’s the basic format:

if(*evaluation*)

{

*statement*;

}

* 1. **SOME EXTRA POINTS**
* whether a function returns a true or false value by reading the function’s documentation, or you can set a true or false return value when writing your own functions.
* You cannot compare strings by using an if comparison. Instead, you use specific string comparison functions.
* When only one statement belongs to an if comparison, the braces are optional.

(08)

**CHAPTER-2**

2. HEADERFILES AND FUNCTION USED

**2.1 HEADERFILES USED**

* #include<stdio.h>
* #include<conio.h>
* #include<process.h>
* #include<windows.h>

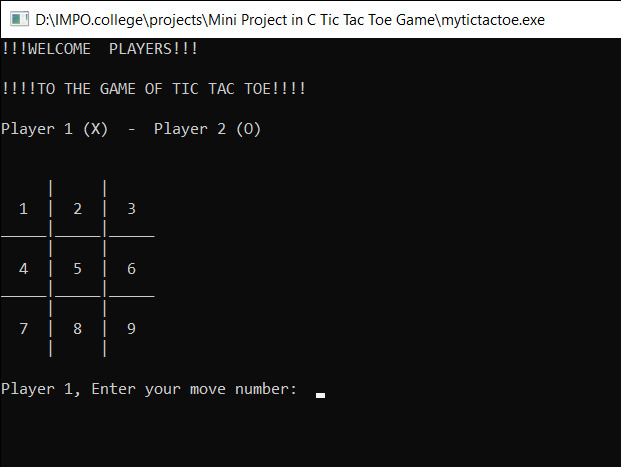
**2.2 FUNCTION USED**

* Void gotoxy();
* int checkwin();
* void board();
* system("cls");

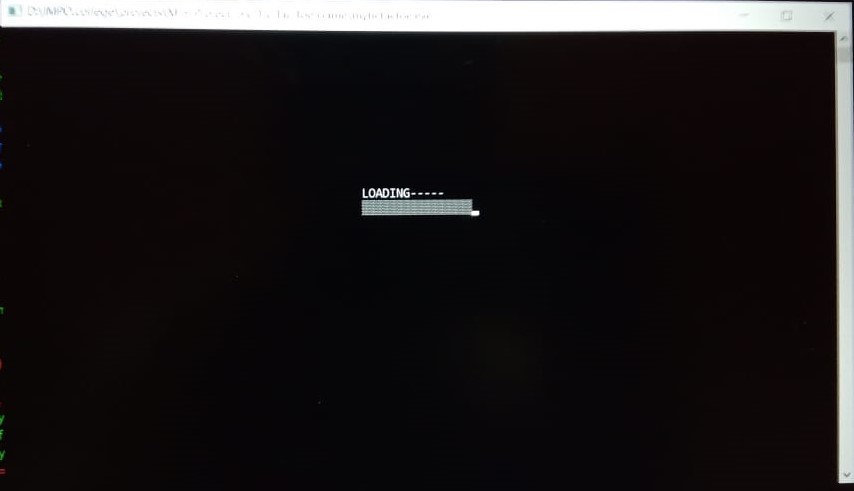
(09)

**5.**  **SNAPSHOTS**

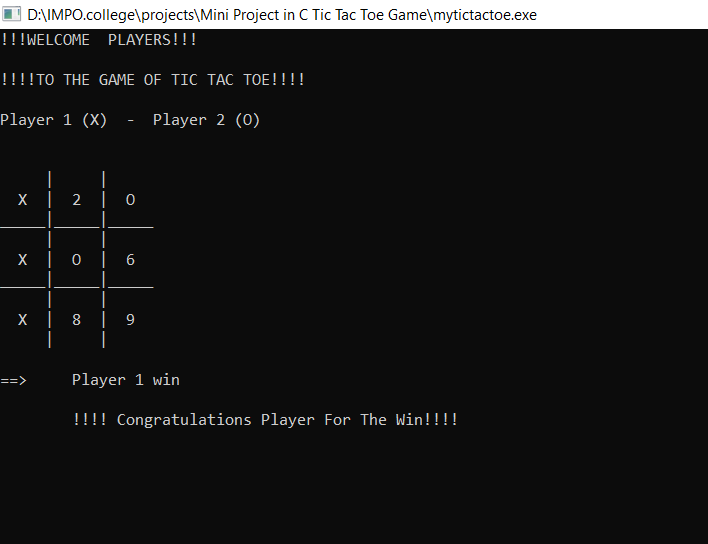
(HERE!! MAIN SCREEN OF THE GAME)



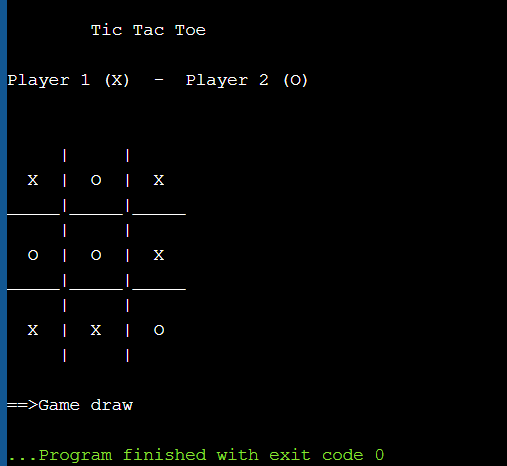
(HERE!! LOADING SCREEN)



(17)

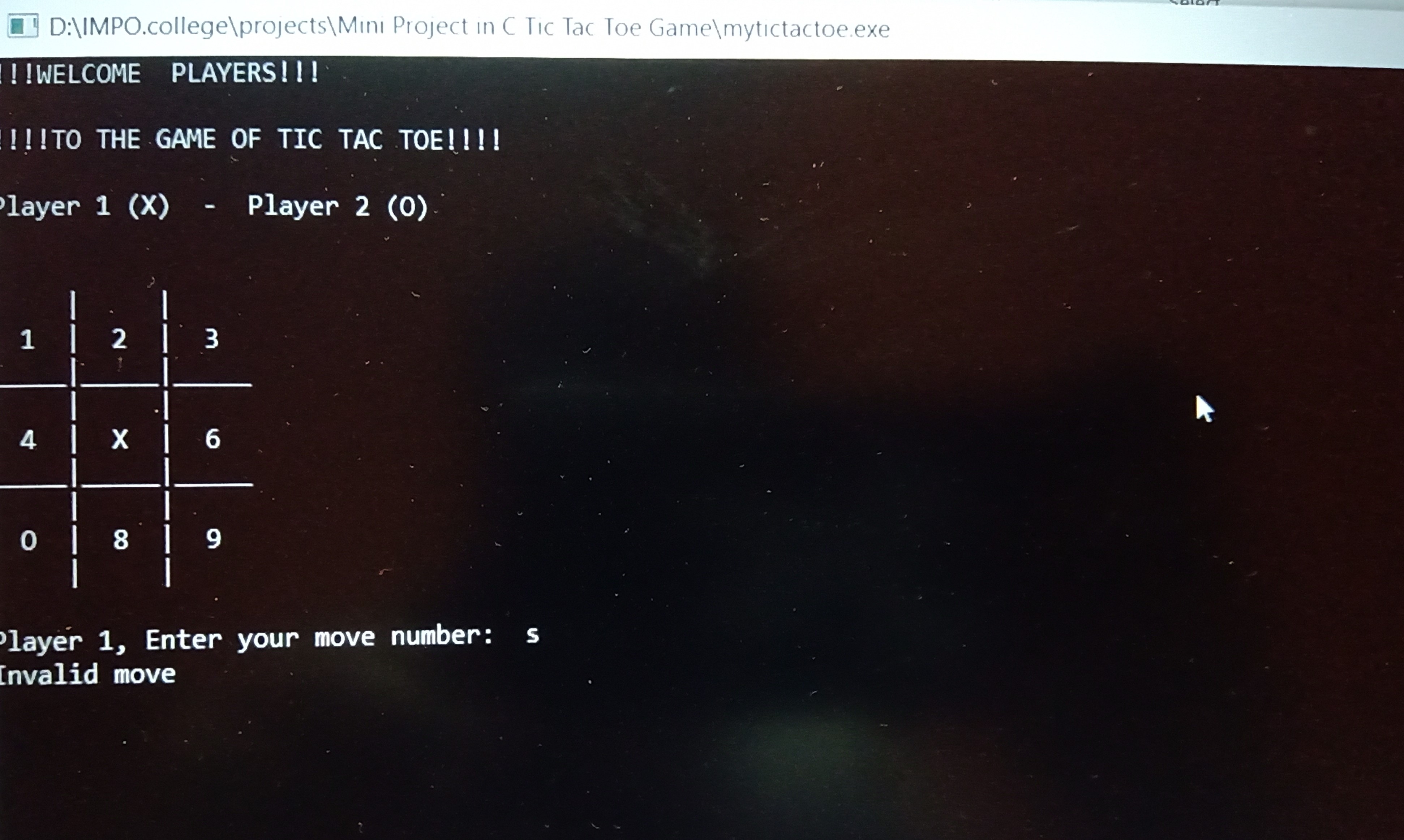


(HERE!!! PLAYER 1 WINS THE GAME)



(HERE!! GAME DRAWS)

(18)



(19)

* 1. **FUTURE SCOPE OF PROJECT**
* Our project will be able to implement in future after making some changes and modifications as we make our project at a very low level. So the modifications that can be done in our project are:

1. It can be made with good graphics.

2. We can set time for each chance

3. We can add different arena in the game.

4. We can give choice to the players to select ‘x’ or ‘o’.

5. We can give sound effects.

(20)

**BIBLIOGRAPHY**

* **BOOKS**
* Learn c programming.
* Let us c.
* **WEBSITES**
* [www.quora.com](http://www.quora.com)
* [www.google.com](http://www.google.com)

(21)