**PSP Course Project**

**Title:** Tic-Tac-Toe Game

**Group No.:** 6

**Section:** A2

**Group Members:**

2103A51058 - Mohammed Nabeel Ahmed

2103A51135 - Mohammed Saif Najmuddin

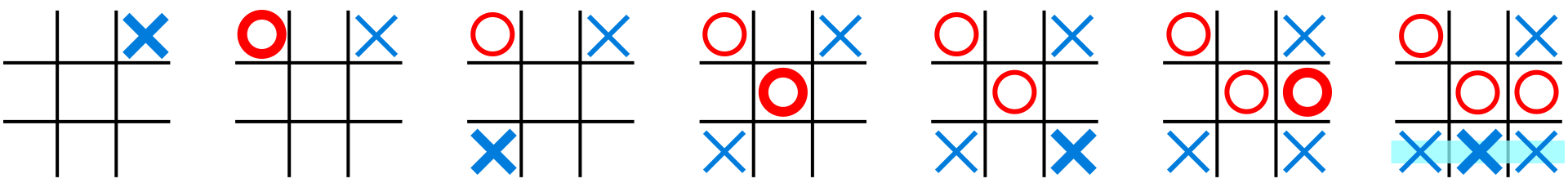
2103A51076 - V. Nandhitha

2103A51552 - B. Bhuwan Rao

**1.Abstract:**

Tic-tac-toe is played on a three-by-three grid by two players, who alternately place the marks X and O in one of the nine spaces in the grid.

In the following example, the first player (*X*) wins the game in seven steps:



There is no universally-agreed rule as to who plays first, but in this article the convention that X plays first is used.

Players soon discover that the best play from both parties leads to a [draw](https://en.wikipedia.org/wiki/Draw_(tie)). Hence, tic-tac-toe is often played by young children who may not have discovered the optimal strategy.

Because of the simplicity of tic-tac-toe, it is often used as a [pedagogical](https://en.wikipedia.org/wiki/Pedagogical) tool for teaching the concepts of good [sportsmanship](https://en.wikipedia.org/wiki/Sportsmanship) and the branch of [artificial intelligence](https://en.wikipedia.org/wiki/Artificial_intelligence) that deals with the searching of [game trees](https://en.wikipedia.org/wiki/Game_tree). It is straightforward to write a [computer program](https://en.wikipedia.org/wiki/Computer_program) to play tic-tac-toe perfectly or to enumerate the 765 essentially different positions (the [state space complexity](https://en.wikipedia.org/wiki/State_space_complexity)) or the 26,830 possible games [up to](https://en.wikipedia.org/wiki/Up_to) rotations and reflections (the [game tree complexity](https://en.wikipedia.org/wiki/Game_tree_complexity)) on this space.If played optimally by both players, the game always ends in a draw, making tic-tac-toe a [futile game](https://en.wikipedia.org/wiki/Futile_game).

Games played on three-in-a-row boards can be traced back to [ancient Egypt](https://en.wikipedia.org/wiki/Ancient_Egypt),where such game boards have been found on roofing tiles dating from around 1300 BC.

An early variation of tic-tac-toe was played in the [Roman Empire](https://en.wikipedia.org/wiki/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 had only 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. Another closely related ancient game is [three men's morris](https://en.wikipedia.org/wiki/Three_men%27s_morris) which is also played on a simple grid and requires three pieces in a row to finish, and [Picaria](https://en.wikipedia.org/wiki/Picaria), a game of the [Puebloans](https://en.wikipedia.org/wiki/Puebloans).

The different names of the game are more recent. The first print reference to "noughts and crosses" ([nought](https://en.wikipedia.org/wiki/Nought) being an alternative word for 'zero'), the British name, appeared in 1858, in an issue of [*Notes and Queries*](https://en.wikipedia.org/wiki/Notes_and_Queries). The first print reference to a game called "tic-tac-toe" occurred in 1884, but referred to "a children's game played on a slate, consisting of trying with the eyes shut to bring the pencil down on one of the numbers of a set, the number hit being scored". "Tic-tac-toe" may also derive from "tick-tack", the name of an old version of [backgammon](https://en.wikipedia.org/wiki/Backgammon) first described in 1558. The US renaming of "noughts and crosses" to "tic-tac-toe" occurred in the 20th century.

In 1952, [*OXO*](https://en.wikipedia.org/wiki/OXO_(video_game)) (or *Noughts and Crosses*), developed by British computer scientist [Sandy Douglas](https://en.wikipedia.org/wiki/Sandy_Douglas) for the [EDSAC](https://en.wikipedia.org/wiki/EDSAC) computer at the [University of Cambridge](https://en.wikipedia.org/wiki/University_of_Cambridge), became one of the first known [video games](https://en.wikipedia.org/wiki/Video_game). 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](https://en.wikipedia.org/wiki/Massachusetts_Institute_of_Technology) students to demonstrate the computational power of [Tinkertoy](https://en.wikipedia.org/wiki/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](https://en.wikipedia.org/wiki/Museum_of_Science,_Boston).

**2.Introduction:**

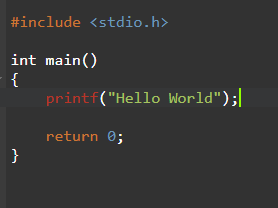
**C Language:**

C is a [general-purpose](https://en.wikipedia.org/wiki/General-purpose_language) computer [programming language](https://en.wikipedia.org/wiki/Programming_language). It was created in the 1970s by [Dennis Ritchie](https://en.wikipedia.org/wiki/Dennis_Ritchie), and remains very widely used and influential. By design, C's features cleanly reflect the capabilities of the targeted CPUs. It has found lasting use in [operating systems](https://en.wikipedia.org/wiki/Operating_system), [device drivers](https://en.wikipedia.org/wiki/Device_drivers), [protocol stacks](https://en.wikipedia.org/wiki/Protocol_stack), though decreasingly for [application software](https://en.wikipedia.org/wiki/Application_software), and is common in computer architectures that range from the largest [supercomputers](https://en.wikipedia.org/wiki/Supercomputer) to the smallest [microcontrollers](https://en.wikipedia.org/wiki/Microcontroller) and [embedded systems](https://en.wikipedia.org/wiki/Embedded_system).

A successor to the programming language [B](https://en.wikipedia.org/wiki/B_(programming_language)), C was originally developed at [Bell Labs](https://en.wikipedia.org/wiki/Bell_Labs) by [Dennis Ritchie](https://en.wikipedia.org/wiki/Dennis_Ritchie) between 1972 and 1973 to construct utilities running on [Unix](https://en.wikipedia.org/wiki/Unix). It was applied to re-implementing the kernel of the Unix operating system. During the 1980s, C gradually gained popularity. It has become one of the [most widely used programming languages](https://en.wikipedia.org/wiki/Measuring_programming_language_popularity), with C [compilers](https://en.wikipedia.org/wiki/Compiler) available for almost all modern [computer architectures](https://en.wikipedia.org/wiki/Computer_architecture) and operating systems. C has been standardised by [ANSI](https://en.wikipedia.org/wiki/American_National_Standards_Institute) since 1989 ([ANSI C](https://en.wikipedia.org/wiki/ANSI_C)) and by the [International Organization for Standardization](https://en.wikipedia.org/wiki/International_Organization_for_Standardization) (ISO).

C is an [imperative](https://en.wikipedia.org/wiki/Imperative_programming) [procedural](https://en.wikipedia.org/wiki/Procedural_programming) language supporting [structured programming](https://en.wikipedia.org/wiki/Structured_programming), [lexical variable scope](https://en.wikipedia.org/wiki/Lexical_variable_scope), and [recursion](https://en.wikipedia.org/wiki/Recursion_(computer_science)), with a [static type system](https://en.wikipedia.org/wiki/Static_type_system). It was designed to be [compiled](https://en.wikipedia.org/wiki/Compiler) to provide [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) access to [memory](https://en.wikipedia.org/wiki/Computer_memory) and language constructs that map efficiently to [machine instructions](https://en.wikipedia.org/wiki/Machine_code), all with minimal [runtime support](https://en.wikipedia.org/wiki/Runtime_system). Despite its low-level capabilities, the language was designed to encourage cross-platform programming. A [standards](https://en.wikipedia.org/wiki/Specification_(technical_standard))-compliant C program written with [portability](https://en.wikipedia.org/wiki/Software_portability) in mind can be compiled for a wide variety of computer platforms and operating systems with few changes to its source code.

A C program typically consists of preprocessor directives, the main() function where the code is written, within the main() function variable declaration,keywords,calling of function etc. are present.



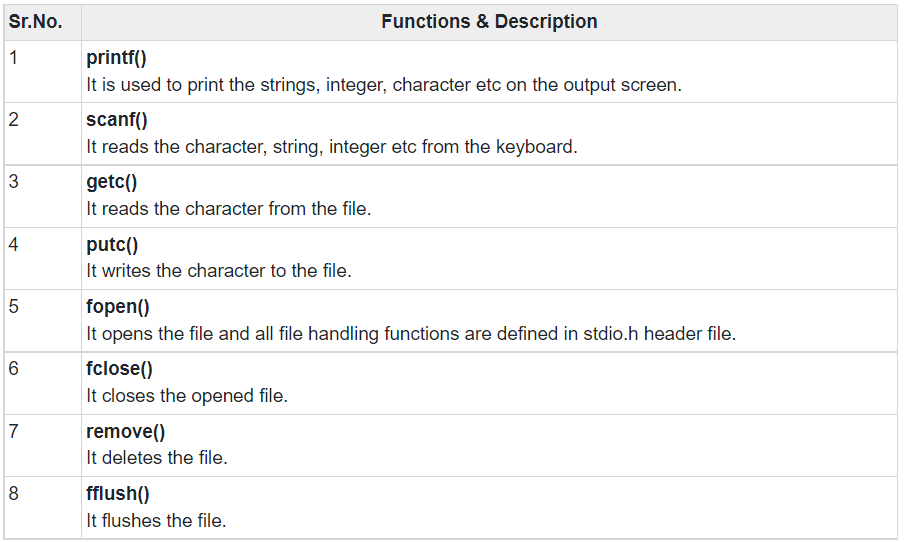
**3.Modules/Libraries included:**

**To make a working Tic-tac-toe game in C language we use two libraries which are included using the preprocessor directive #include.**

**1.Standard Input/Output header file <stdio.h>:**

The stdio.h stands for standard input/output header file. It has information related to Input/Output functions.

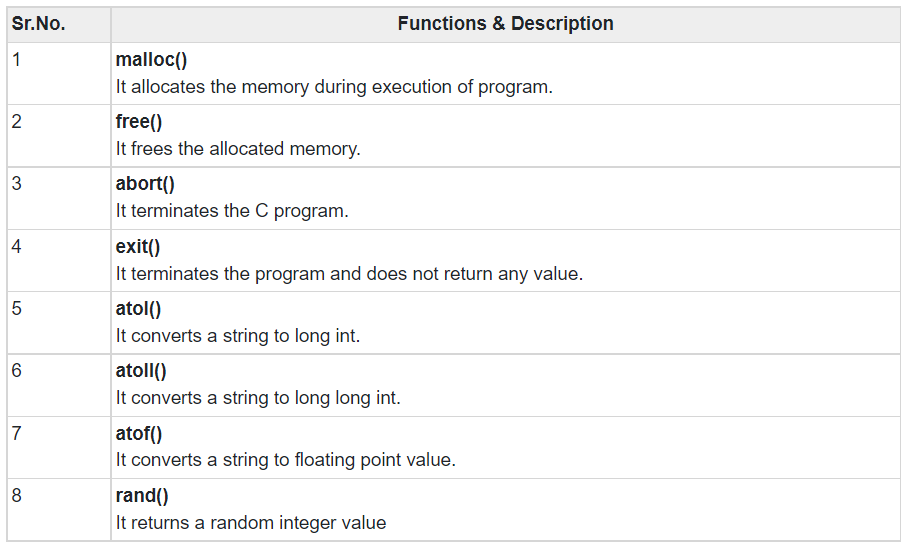
Here is the table that displays some functions in stdio.h in C language,



**2.standard library header file <stdlib.h>:**

The stdlib.h stands for standard library header file. It has the information of memory allocation/freeing functions. To make a tic-tac-toe game we particularly use the system() function to clear the output window.

Here is the table that displays some functions in stdlib.h in C language,



* **system() function in <stdlib.h>:**

The system() is a part of the C standard library. It is used to pass commands that can be executed in the command processor or the terminal of the operating system, and finally returns the command after it has been completed.

**Syntax:**

****

**Example:**

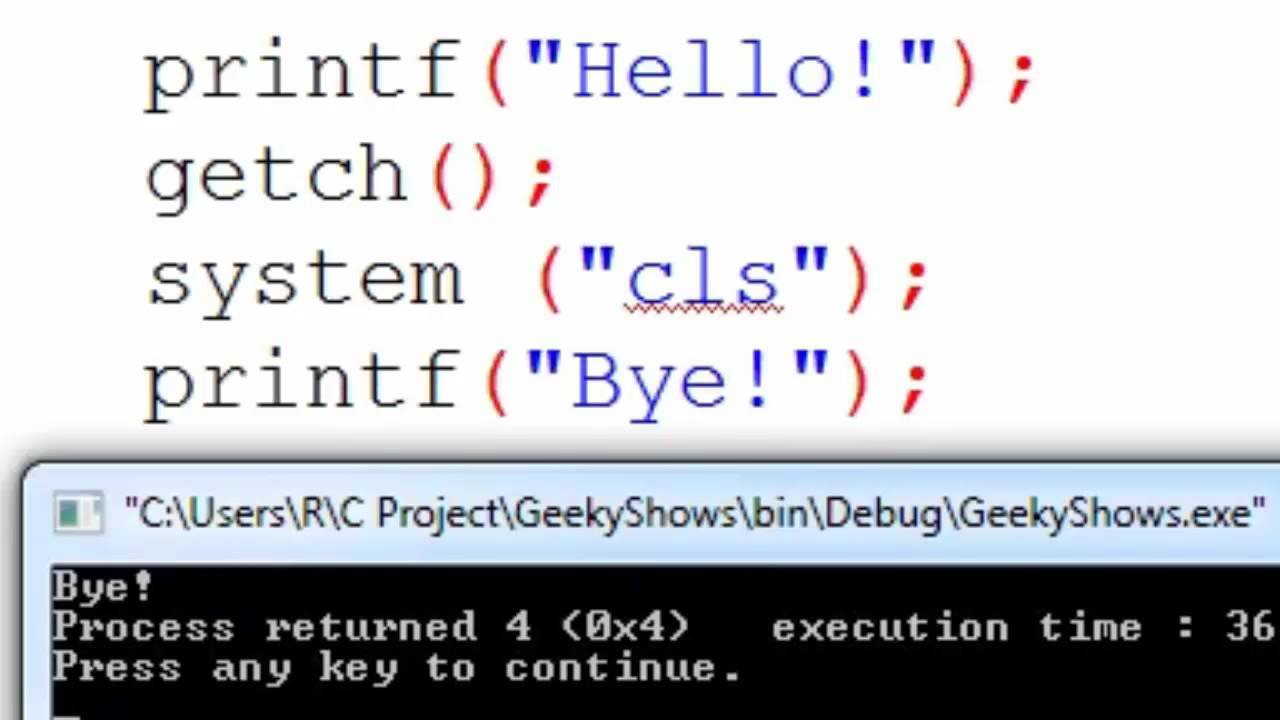
****

****

**Using system(“cls”); in C:**

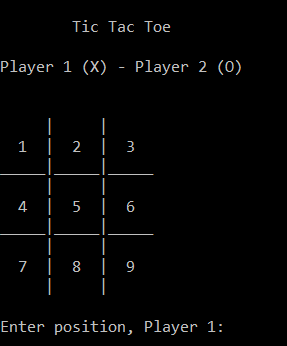
This command is used to wipe out the output window.

Example of system(“cls”); in C:



**4.Implementation:**

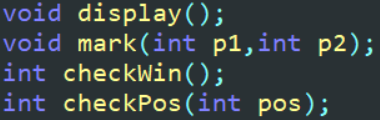
The implementation of a working Tic-tac-toe game in C starts with an idea of displaying a game-board with positions marked in it. The players have to choose one of the unoccupied positions to mark their respective symbol (either X or O) on the game-board.



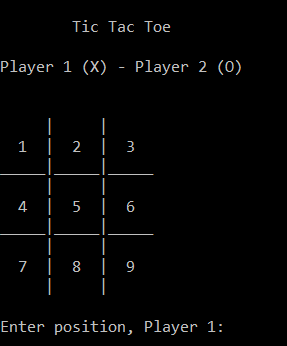
To make a game board we declare a character array globally so that we can access it and make changes through every function. An alternative for this can be using memory allocations locally as the pointer array can be accessed and changed through every function.



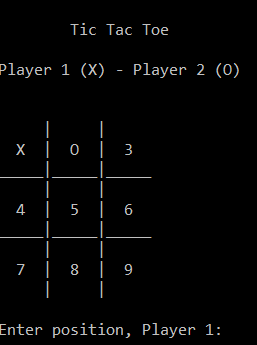
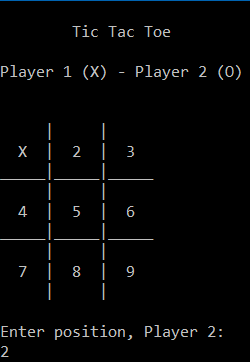
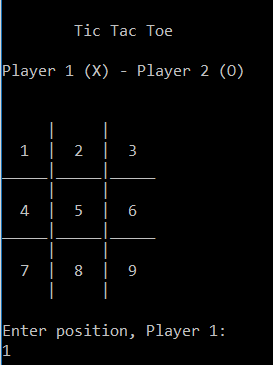
Then we use 4 different functions:



1. **void display();** function is used to display the game-board in the output window.

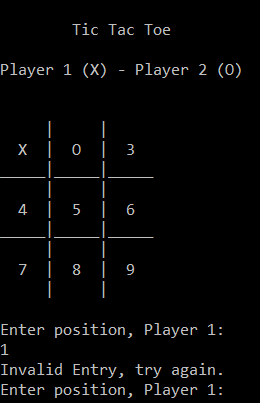


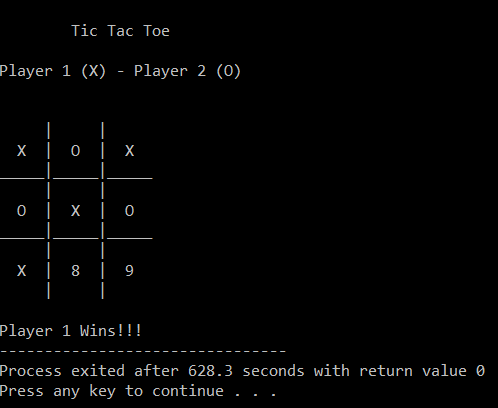
1. **Void mark(int p1, int p2);** function is used to mark the symbol (either X or O) in an unoccupied position by taking an input from the player. The arguments p1 and p2 are the input positions from the players.



Hence we use a loop and ask the players to input their desired positions.

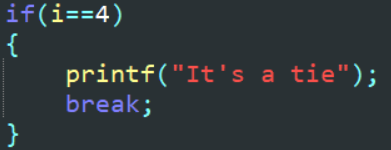
1. **int checkPos(int pos);** function is used to check if a player has given an already occupied position as input. It returns a boolean value and asks the same player to enter their desired position again.

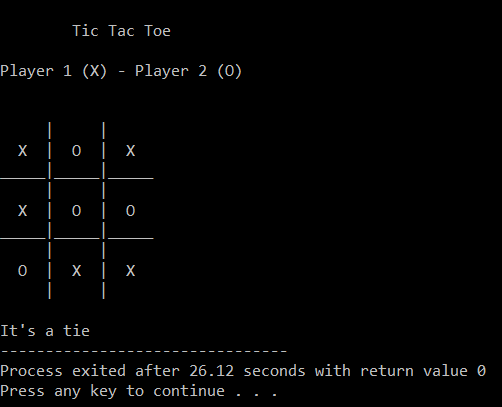


1. **int checkWin();** function checks if a player has won or not. It returns a boolean value and breaks the loop telling that a player has won.

* **The tie:**

To check if the game is a tie between the two players we check if the iteration in the for loop has reached its limit. If the limit of iteration has reached then the loop breaks telling that it is a tie.





* **system(“cls”)** in the mark() function to clear the output window every time a player enters a position and re-display the game-board with the symbol marked using display() function.



**5.Test Results:**

**The final code:**

#include <stdio.h>

#include <stdlib.h>

char p[10]={'0','1','2','3','4','5','6','7','8','9'}; //This array represents a game-board.

void display(); //This function displays the game-board.

void mark(int p1,int p2); //This function assigns X or O to the given input position.

int checkWin(); //This function checks if a player has won or not.

int checkPos(int pos); //This function checks the input position.

main()

{

int i,p1,p2;

char play;

display();

GAME:

for(i=0;i<5;i++)

{

PLAYER\_1:

printf("Enter position, Player 1:\n");

scanf("%d",&p1);

if(checkPos(p1)==1)

{

printf("Invalid Entry, try again.\n");

goto PLAYER\_1;

}

mark(p1,p2);

if(checkWin()==1)

{

printf("Player 1 Wins!!!");

break;

}

if(i==4)

{

printf("It's a tie");

break;

}

PLAYER\_2:

printf("Enter position, Player 2:\n");

scanf("%d",&p2);

if(checkPos(p2)==1)

{

printf("Invalid entry, try again.\n");

goto PLAYER\_2;

}

mark(p1,p2);

if(checkWin()==2)

{

printf("Player 2 Wins!!!");

break;

}

}

}

void display()

{

printf("\n\tTic Tac Toe\n\n");

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

printf(" | | \n");

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

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

printf(" | | \n");

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

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

printf(" | | \n");

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

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

}

void mark(int p1,int p2)

{

switch(p1)

{

case 1:

p[1]='X';

break;

case 2:

p[2]='X';

break;

case 3:

p[3]='X';

break;

case 4:

p[4]='X';

break;

case 5:

p[5]='X';

break;

case 6:

p[6]='X';

break;

case 7:

p[7]='X';

break;

case 8:

p[8]='X';

break;

case 9:

p[9]='X';

break;

}

switch(p2)

{

case 1:

p[1]='O';

break;

case 2:

p[2]='O';

break;

case 3:

p[3]='O';

break;

case 4:

p[4]='O';

break;

case 5:

p[5]='O';

break;

case 6:

p[6]='O';

break;

case 7:

p[7]='O';

break;

case 8:

p[8]='O';

break;

case 9:

p[9]='O';

break;

}

system("cls"); //To refresh the output window.

display(); //To display the game-board again with X and O.

}

int checkWin()

{

int returnValue=0;

//If any row,column or diagonal contains same character returnValue=1

if(p[1]=='X'&&p[2]=='X'&&p[3]=='X')

returnValue=1;

else if(p[4]=='X'&&p[5]=='X'&&p[6]=='X')

returnValue=1;

else if(p[7]=='X'&&p[8]=='X'&&p[9]=='X')

returnValue=1;

else if(p[1]=='X'&&p[4]=='X'&&p[7]=='X')

returnValue=1;

else if(p[2]=='X'&&p[5]=='X'&&p[8]=='X')

returnValue=1;

else if(p[3]=='X'&&p[6]=='X'&&p[9]=='X')

returnValue=1;

else if(p[1]=='X'&&p[5]=='X'&&p[9]=='X')

returnValue=1;

else if(p[3]=='X'&&p[5]=='X'&&p[7]=='X')

returnValue=1;

//---------------------------------------------

else if(p[1]=='O'&&p[2]=='O'&&p[3]=='O')

returnValue=2;

else if(p[4]=='O'&&p[5]=='O'&&p[6]=='O')

returnValue=2;

else if(p[7]=='O'&&p[8]=='O'&&p[9]=='O')

returnValue=2;

else if(p[1]=='O'&&p[4]=='O'&&p[7]=='O')

returnValue=2;

else if(p[2]=='O'&&p[5]=='O'&&p[8]=='O')

returnValue=2;

else if(p[3]=='O'&&p[6]=='O'&&p[9]=='O')

returnValue=2;

else if(p[1]=='O'&&p[5]=='O'&&p[9]=='O')

returnValue=2;

else if(p[3]=='O'&&p[5]=='O'&&p[7]=='O')

returnValue=2;

return returnValue;

}

int checkPos(int pos)

{

int returnValue=0;

switch(pos) //switch checks if input position is re-entered.

{

case 1:

if(p[1]!='1')

returnValue=1;

break;

case 2:

if(p[2]!='2')

returnValue=1;

break;

case 3:

if(p[3]!='3')

returnValue=1;

break;

case 4:

if(p[4]!='4')

returnValue=1;

break;

case 5:

if(p[5]!='5')

returnValue=1;

break;

case 6:

if(p[6]!='6')

returnValue=1;

break;

case 7:

if(p[7]!='7')

returnValue=1;

break;

case 8:

if(p[8]!='8')

returnValue=1;

break;

case 9:

if(p[9]!='9')

returnValue=1;

break;

default: //defult checks if the input position is less then 1 or greater than 9.

returnValue=1;

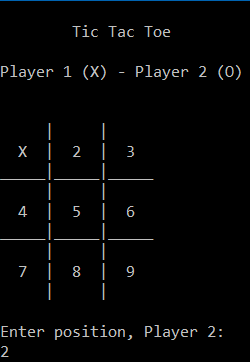
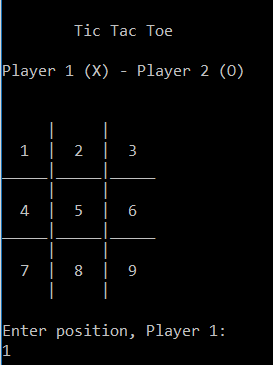
}

return returnValue;

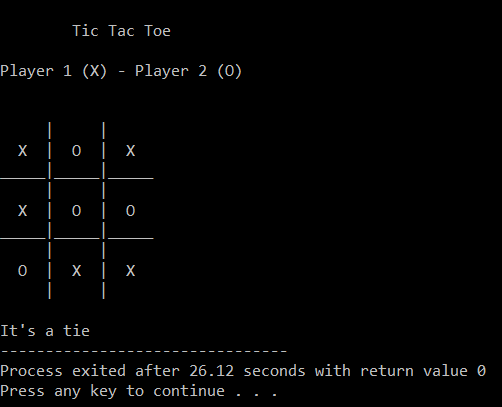
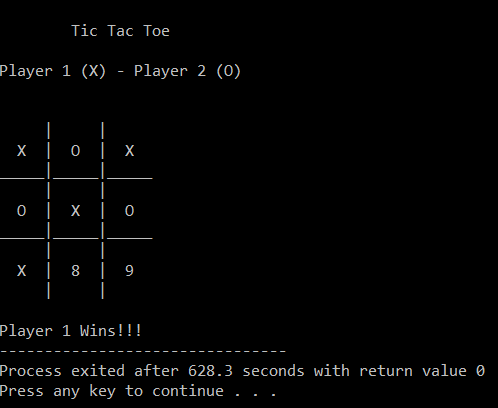
}

**The output:**

The players are required to choose their desired position and continue playing the game.



After every input the game-board is updated.



If a player wins or if it is a tie then it is displayed at the end.

**6.Conclusion and Future scope:**

* It is possible and quite simple to make a working Tic-tac-toe game in C.
* Alternatively using pointers this game can be made.
* The same kind of logic can be used to make applications like movie ticket booking, calendar etc.
* This program can be upgraded by using a database and storing scores of each player.
* Using the standard library we can make changes in the output window and it is very useful.