**Tic-Tac-Toe Gameplay Portal**

## Submitted by

**Name of the Students:** Sneha Singh

**Enrolment Number:** 12022002011067

**Section:** I

**Class Roll Number:** 72

**Stream:** EE

**Subject:** Programming for Problem Solving

**Subject Code:** ESC103(Pr)

**Department:** Basic Science and Humanities

**Academic Year: 2022-26**

#### PROJECT REPORT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE SECOND SEMESTER

Under the supervision of Prof. Swarnendu Ghosh Prof. Dr. Sumana Sinha



**DEPARTMENT OF BASIC SCIENCE AND HUMANITITES INSTITUTE OF ENGINEERING AND MANAGEMENT, KOLKATA**



## CERTIFICATE OF RECOMMENDATION

#### We hereby recommend that the project prepared under our supervision by **,** entitled Tic-Tac-Toe Gameplay Portal be accepted in partial fulfillment of the requirements for the degree of partial fulfillment of the first semester.

\_

#### Head of the Department Project Supervisor Basic Sciences and Humanities

IEM, Kolkata

# Introduction

The age old tic-tac-toe game which everyone has played from generation to generation had and has always been everyone’s favorite easy indoor game. Tic-tac-toe is a game requiring a little tactics but is very fun to play.

Features of a tic-tac-toe game are :-

* 1. It is a game played by etching a matrix of 3 by 3.
  2. It is a two-player game.
  3. It is normally played in offline mode but now played in mobile devices as well.
  4. It is played with the help of two symbols i.e., ‘X/x’ or ‘O/o’.
  5. Which player will make the first move is mutually decided among the two and who’ll take which sign is also decided mutually.

Rules of Tic-tac-toe game are :-

1. Each player chooses a sign either cross (X/x) or Circle (O/o) and continues to put a mark with that only as the game proceeds.
2. In each cell out of the 9 cells each player marks it with their chosen symbol one by one in turn.
3. If one symbol of a player forms a straight line pattern across any three cells either in a horizontal /vertical /diagonal line.
4. The player whose symbol forms a continuous line wins the game.
5. While the game-result can also be a TIE if-
   1. Both the players’ symbol form a continuous line.
   2. Neither of them has been able to form a continuous line with their symbol.

This game is easy and fun to play, wherein players also have to put in a little tactics so as to win it at last.

The program designed here provides player with all the instructions and all possible options to choose from –

1. Who would start the game ?
2. Which sign would the player choose?
3. Position at which player will put the mark. (iv). Scores of each player.
4. Results of the Game.
5. Does the user want to play another game ?

The program is divided into several modular functions each serving and solving its specific purpose while being called from the main-function.

So, a program has been created using C language to create a 2-player model for the game Tic-Tac-Toe keeping in mind all the features and rules of the game to

offer the players a new gameplay experience by enabling options for choosing player- no., sign-choice and entering position- coordinates.

After making each move the players would be able to view the checkboard to make their next move in a calculative manner.

At the end of the 9th move, the checkboard would be displayed finally along with Final scores for individual player and thereby declaring final result of the Game.

1. **Detailed Description of the Gameplay Portal**
   1. **Objective**

This gameplay portal prompts the user with instruction at each point of the game and accordingly proceeds the game otherwise returns to the user with a request to once again input their choice. This program aims to efficiently:-

* + 1. Take user-choice for starter-player and remember which player would make the first move.
    2. Ask the starter-player his/her choice of Symbol and accordingly assign the starter player with his/her choice of symbol and the other player with the opposite symbol.
    3. Constantly ask each user to input their desired position to mark their symbol, from position numbers: 1 to 9.
    4. At the end of all markings the checkboard is thoroughly checked for all possible combinations of continuous-line patterns with same symbol and assign score to the player in possession with symbol for that game.
    5. Display Score of each individual player and Declaring Result of the game and asking the users if they want to play another game.

##### Organization of the Project

* + 1. **This program fetches the following data from the user:-**

(i). Take user-choice for starter-player either: player-no. ‘1’ / ‘2’; (ii). Take sign-choice for starter-player either: ‘X/x’ or ‘O/o’;

* + - 1. Ask the starter-player and then the other player who would make the next following move via entering position numbers on the 3x3 matrix: (1/2/3/4/5/6/7/8/9);
      2. To ask at the end of the game whether the user wants to play Again.

##### Functions & labels list and description with scope & data-type:

###### int main() -

* to take user-input;
* to verify them at valid checkpoints;
* to assign players their desired symbol;
* keep track of no. of moves made
* keep track of turn of which player
* call necessary function. (ii). ***gameplay:*** -
* to define block of statement recalled at New-game instance. (iii). ***playerchoice:*** -
* to define block of statement recalled at Wrong starter player-no. input. (iv). ***signchoice:*** -
* to define block of statement recalled at Wrong input for sign-choice by starter player.

###### int mark(int, char) -

* to verify input at valid checkpoints;
* to mark gameplay-checkboard according to player-input with assigned player- sign;
* to decrement move-no. and reassign player-no. at wrong input and return to main function.

###### void view(int) -

* to display gameplay-checkboard after each move is made;
* if a position is left unmarked display it with “!”. (vii). ***void scrchk(char, char)*** -
* called at end of 9th move to check individual player score;
* to thoroughly traverse the whole display gameplay-checkboard;
* check for marking of same sign for all possible combinations of continuous line-pattern;
* to increment scores accordingly by calling *adder();*
* print score of individual player. (viii). ***void adder(char)*** -
* to increment score of player with assigned symbol who has checked to form a continuous line-pattern by *scrchk()*.

(ix). ***int chkrslt(int, int)*** -

* called after *scrchk();*
* to compare scores of the two players;
* to display final result of the gameplay.

##### Variables list and description with scope, purpose, data-type & format- specifier:

1. Starter-Player no. – in main- function within label: playerchoice:

**plyr** (int, % d, options to enter -1 /2 );

1. Current player-no. – in global-scope: ***pl*** *(int, % d, initialized with plyr)*;
2. Sign-choice by starter-player – in main-function within label: signchoice: ***ch*** *(int, % d, options to enter – 1* =*X / 2 =x / 3 =O / 4=o )*;
3. Counter for Move-no. as player- instance – in global-scope: ***pi***

(initialized with 1 in label: gameplay);

1. Signs for player #1 & #2 – [assign each player their desired sign *(X / O)*]: in global-scope: ***s1***, ***s2*** *(char, %c, uninitialized)*.
2. Position-no.- [for marking the player-sign on checkboard]: in global-scope:

**n** (int, % d, uninitialized).

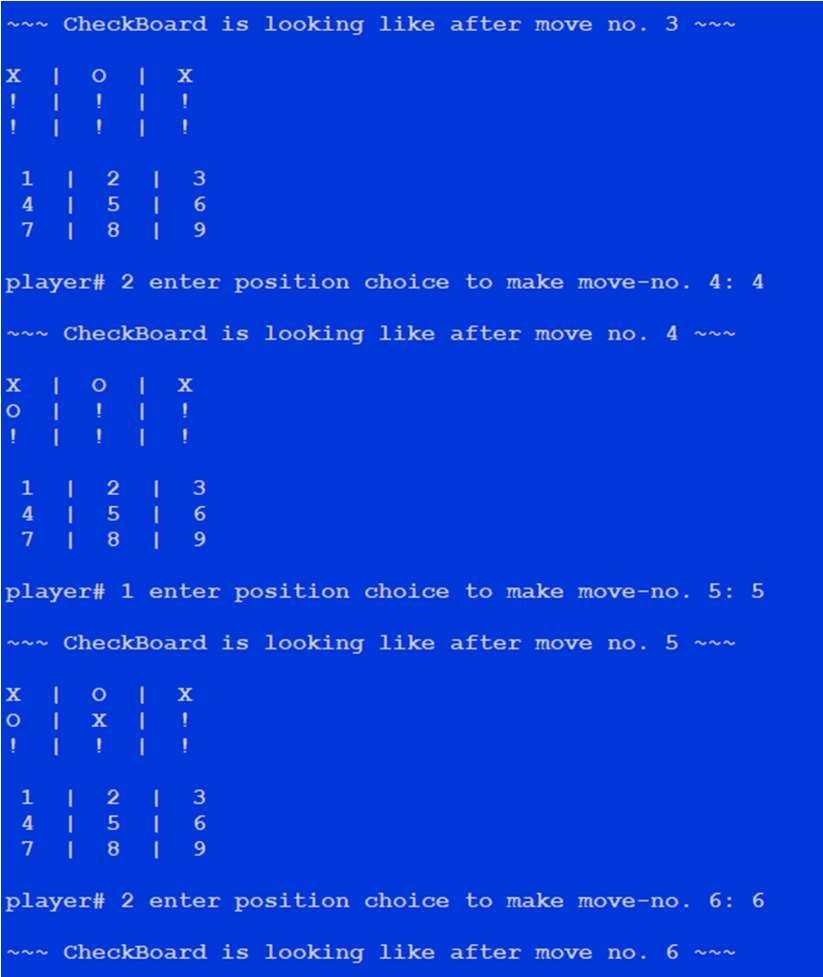
1. Sign of current-player – in global-scope: ***s*** *(char, %c, overwritten with current-player symbol).*
2. Player-scores – [keep track of scores of each player] in global- scope: ***scr1***, ***scr2*** *(int, % d, initialized with 0 in label: gameplay).*
3. Another-game – in global-scope: ***agn*** *(int, % d, uninitialized).*
4. Position-markers – [matrix-index traversing variables]: in mark function: ***i*** (=row), ***j*** (=column), ***k*** (=matrix-level increment) *(int,*

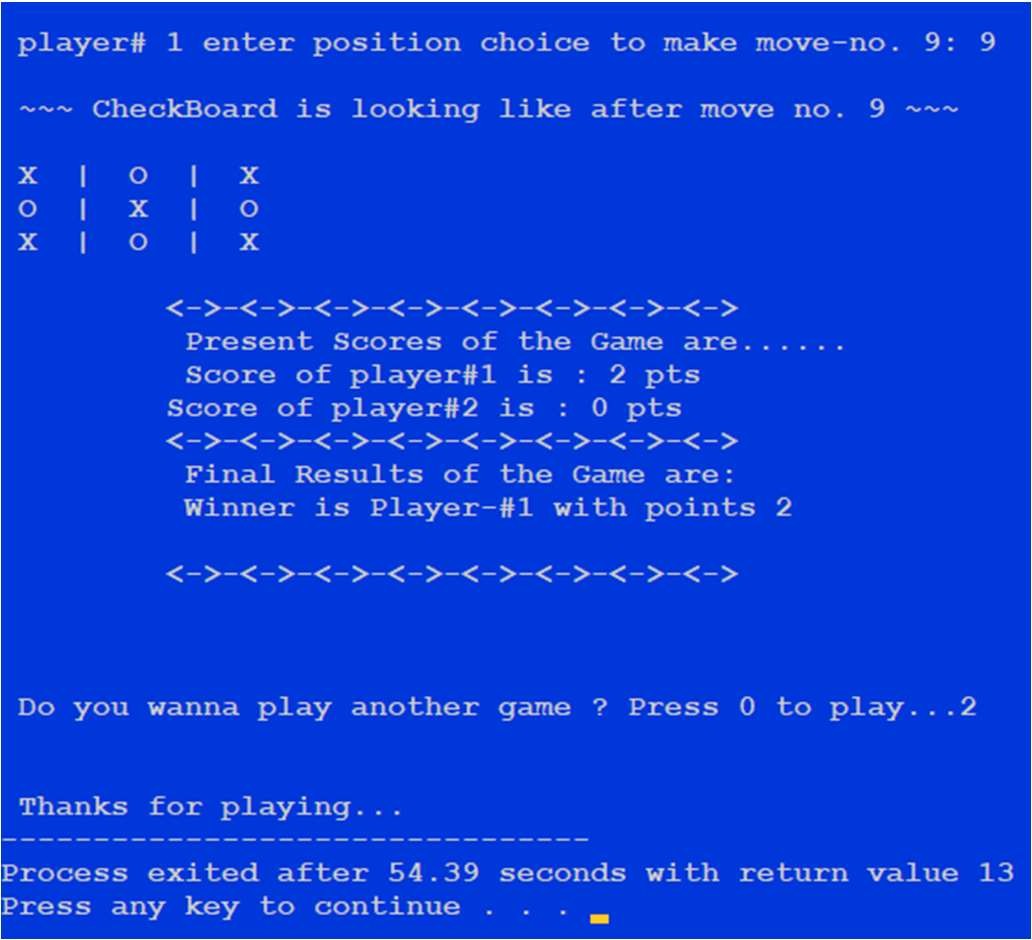
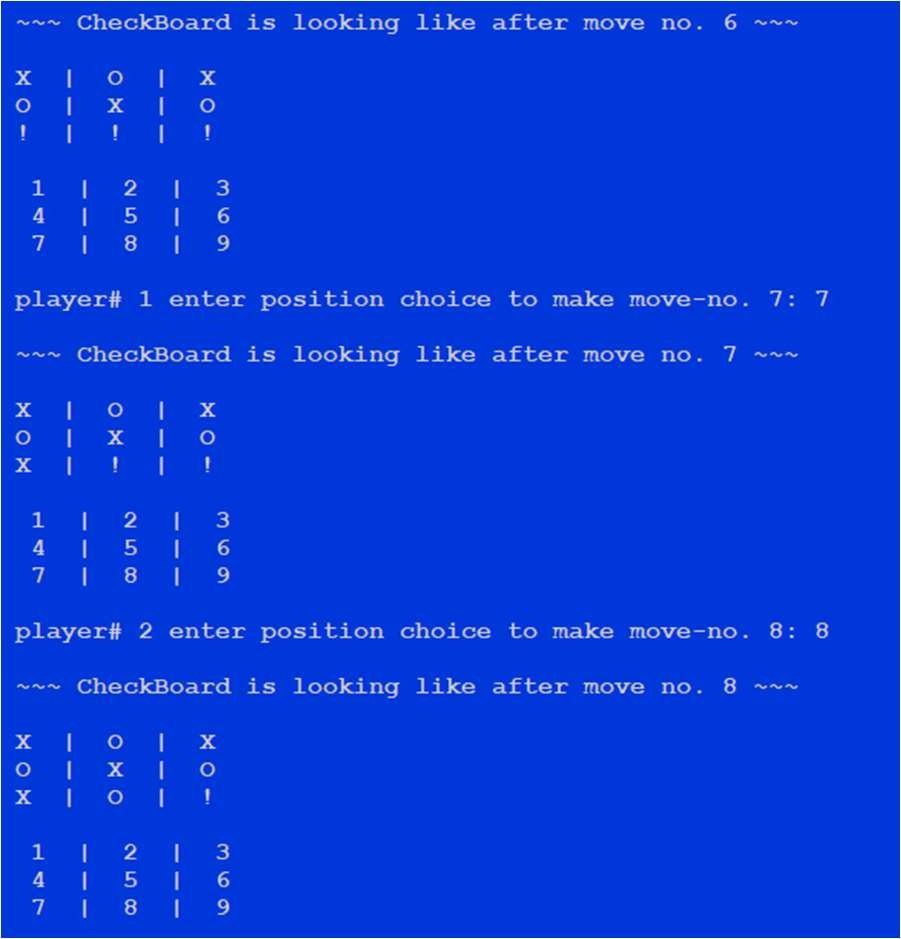
%d, uninitialized).

1. Gameplay-checkboard – 3x3 character matrix – in global-scope: ***a[3][3]*** *(char, %c, written with symbol of current-player* ***s*** *at each new board-instance of* ***pi,*** *reset to NULL with new game in label: gameplay ).*
2. View Position-markers – [matrix-index traversing variables]: in view function: ***l*** (=row), ***m*** (=column) *(int, % d, uninitialized).*
3. Score-check Position-markers – [matrix-index traversing variables]: in scrchk function: ***x*** (=row), ***y*** (=column) *(int, % d, uninitialized).*
4. Sign of current-index – in scrchk function: ***sgn*** *(char, %c, overwritten with symbol of current-index of gameplay matrix).*
5. **Programs**

# Outputs

### Game1



### Game2

