PROJECT REPORT

on

TIC - TAC - TOE

(CSE III Semester Mini project)

2020-2021

Submitted to:

Mr. Umang Garg

(CC-CSE-A-III-Sem)

Guided by:

Dr. Ashook Sahoo

(Resource Person)

Submitted by:

Mr. Pranshu Kukreti

Roll. No.: 1918549

CSE-A-III-Sem

Session: 2020-2021

DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

GRAPHIC ERA HILL UNVERSITY, DEHRADUN

CERTIFICATE

Certified that Mr. Pranshu Kukreti (Roll No. - 1918549) has developed mini project on “Tic-Tac-Toe” for the CS III Semester Mini Project Lab in Graphic Era Hill University, Dehradun. The project carried out by students is their own work as best of my knowledge.

Date:

(Mr. Umang Garg)

Project Co-ordinator

CC-CSE-A-III-Sem

(CSE Department)

GEHU Dehradun

ACKNOWLEDGMENT

We would like to express our gratitude to The Almighty Shiva Baba, the most Beneficent and the most Merciful, for completion of project.

We wish to thank our parents for their continuing support and encouragement. We also wish to thank them for providing us with the opportunity to reach this far in our studies.

We would like to thank particularly our project Co-ordinator Mr. Umang Garg and our Project Guide Dr. Ashook Sahoo for his patience, support and encouragement throughout the completion of this project and having faith in us.

We also acknowledge to teachers who helped us in developing the project.

At last but not the least, we are greatly indebted to all other persons who directly or indirectly helped us during this work.

Mr. Pranshu Kukreti

Roll No. - 1918549

CSE-A-III-Sem

Session: 2020-2021

GEHU, Dehradun

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 ABOUT PROJECT

The aforementioned project undertaken by me is to create a simulated gameplay version of TIC-TAC-TOE, a popular pen-paper game which is a huge hit among the students and adults alike, usually played for enjoyment during the free time for leisure, as well as mind-sharpening and competitive gameplay, on the likes of more intensive board games like GO, Chess, Mahjong, Minesweeper, and many more. It is also called as noughts and crosses or X's and O's. The players take turns marking the spaces in 3x3 grid. The player who succeeds in placing three of their marks in a horizontal, vertical or diagonal row is the winner. It is a solved game with a forced draw assuming best play from both players. If the board is full or when there is not a single slot in the grid, and when neither player has managed to mark a winning row, then it is called a tie. The game is considered as over either when there is a winner, or when there is a tie. The player plays their chances one after another. The part of the grid which is already occupied cannot be overwritten by another player.

1.2 C LANGUAGE

C is a procedural programming language. It was initially developed by Dennis Ritchie in the year 1972 at Bell Labs, USA to develop the UNIX operating system. It was mainly developed as a system programming language to write an operating system. It keeps fluctuating at number one scale of popularity along with Java programming language, which also equally popular and most widely used among modern software programmers. 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 programming like an operating system or compiler development.

1.2.1 DATA TYPES

Integer: Keyword used for integer data types is "**int"**. Integers typically requires 4 bytes of memory space and ranges from -2147483648 to 2147483647.

Character: Character data type is used for storing characters. Keyword used for character data type is "**char"**. Characters typically requires 1 byte of memory space and ranges from -128 to 127 or 0 to 255.

Floating Point: Floating Point data type is used for storing single precision floating point values or decimal values. Keyword used for floating point data type is **"float"**. Float variables typically requires 4 byte of memory space.

Double Floating Point: Double Floating Point data type is used for storing double precision floating point values or decimal values. Keyword used for double floating point data type is **double**. Double variables typically requires 8 byte of memory space.

1.2.2 LOOPS

A **for** loop is a repetition control structure which allows us to write a loop that is executed a specific number of times. The loop enables us to perform 'n' number of steps together in one line.

While loops are used in situations where we do not know the exact number of iterations of loop beforehand. The loop execution is terminated on the basis of test condition.

In **do while** loops also the loop execution is terminated on the basis of test condition. The main difference between do while loop and while loop is in do while loop the condition is tested at the end of loop body, i.e do while loop is exit controlled whereas the other two loops are entry controlled loops.
Note: In do while loop, the loop body will execute at least once irrespective of test condition.

1.2.3 FUNCTIONS

A function is a complete block of code which only runs when it is called. You can pass data, known as parameters, into a function. Functions are used to perform certain actions, and they are important for reusing code: Define the code once, and use it many times.

1.2.4 SWITCH CASE

Use the switch statement to select one of many code blocks to be executed.

- The switch expression is evaluated once
- The value of the expression is compared with the values of each case

- If there is a match, the associated block of code is executed
- The break and default keywords are optional.

1.2.5 ARRAYS

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value. You access an array element by referring to the index number. In array each element is stored in contiguous memory location.

1.2.6 CONDITIONAL STATEMENTS

C++ has the following conditional statements:

- Use if to specify a block of code to be executed, if a specified condition is true
- Use else to specify a block of code to be executed, if the same condition is false
- Use else if to specify a new condition to test, if the first condition is false
- Use switch to specify many alternative blocks of code to be executed

1.2.7 OPERATORS

A list of all comparison operators:

Operator	Name	Example
==	Equal to	x == y
!=	Not equal to	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Tab 1.2 a

Logical operators are used to determine the logic between variables or values:

Operator	Name	Description	Example
&&	Logical AND	Returns true if both statements are true	x < 5 && x < 10
||	Logical OR	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical NOT	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

Tab 1.2 b

CHAPTER 2

PROJECT

2.1 REQUIREMENT ANALYSIS

Minimum Hardware Requirements

- x86, x86_64, ARM, SPARC, RISC-V, or any CISC/RISC based Microprocessor
- 512 MB main memory, i.e., RAM
- 1Mb secondary memory, i.e., Hard Disk
- Display monitor
- keyboard

2.2 SOFTWARE SPECIFICATION

- Basic Operating System, i.e, Windows, Macintosh OS, Linux distributions, BSD, UNIX, Tails OS, DOS, Xerox OS, or any other.
- Installed C/C++ compiler, preferably GNU/G++ or Clang/LLVM or Mingw.
- Installed Shell prompt and dependencies, preferably PowerShell or cmd for Windows/.NET systems, bash/zsh for Linux, BSD, and Mac OS.
- System drivers for Display-out, and keyboard-in, as well as proper input/output channel calibration on a system-wide scale.

2.3 MODULE

2.3.1 Entry of the book *Head First C :: O'Reilly Publication, David Griffiths, Dawn Griffiths, ISBN :: 978-1-449-39991-7*, used to gain basic knowledge of functions, structure of program and their know-how.

2.3.2 Entry of the code file**,** *TicTacToe.c* which is the implemented file in C using the fundamentals learned in the book hitherto mentioned.

2.3.3 Entry of the book *The Complete Reference 4th Edition is Herbert Schildt, ISBN ::*

978-0070411838, used to develop a deeper sense of understanding about its library and header files and other procedural oriented programming realm.

2.3.4 Entry of the Project Synopsis file, named *Project Synopsis.pdf.*

2.3.5 Entry of the Project Folder *TIC-TAC-TOE.*

CHAPTER 3

SNAPSHOT OF PROJECT

3.1 TAKING INPUT

The first task is to enter the name of both players who want to play the game. We enter the name of players and assign the names to the X player & O player. The length of the player name should not be large (not greater than 20), otherwise an error prompt will be seen and the program will terminate. Thereafter, the program asks for entering the name of player which we want to play first.

Then we call the function accordingly and make user to enter alternate 'X' and 'O' till we get a result of winner or a draw (tie).

Then we print the result of the game.

```
void main()
{
    char board[]={'1','2','3','4','5','6','7','8','9'};
    printf("\n\n\n----------WELCOME TO TIC-TAC-TOE----------");
    printf("\n\nEnter the number to mark your chance (X/O):\n\n");
    showboard(board);

    //declaring the string
    char firstplayer[21],xplayer[21],oplayer[21];
    printf("\n\nEnter the nickname of X player: ");
    scanf("%[^\n]s",xplayer);
    fflush(stdin);
    printf("Enter the nickname of O player: ");
    scanf("%[^\n]s",oplayer);

   if((strlen(xplayer)>20)||(strlen(oplayer)>20))
    {
        printf("\nERROR!! Enter a smaller nickname.\n TRY AGAIN!!");
        exit(1);
    }
    int marker,checkwinner=0, boardfull=0, flag=0, randomnumber=0;
    fflush(stdin);
    printf("Enter the name of the player who want to play first:");
    scanf("%[^\n]s",firstplayer);
    if(strcmp(firstplayer,oplayer)==0){
        randomnumber=0;
    }
    else if(strcmp(firstplayer,xplayer)==0){
        randomnumber=1;
    }
    else{
        printf("The player name: %s does not exist.",firstplayer);
        exit(1);
    }
```

Fig. 3.1 a

```c
for(int i=1;i<=18;i++)
{
    if(randomnumber==0)
    {
        printf("\n%s's turn(O): ",oplayer);
        fflush(stdin);
        scanf("%d",&marker);
        system("cls");  //CLEAR THE WHOLE SCREEN
        if(marker==1){
                if(board[0]=='O'||board[0]=='X'){
                    randomnumber=1;
                    printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                    Sleep(2000);   //THE PROGRAM SLEEPS FOR THE SPECIFIED TIME AND CONTINUES FROM THE EXISTING PLACE
                    showboard(board);
                    continue;
                }
                board[0]='O';
        }
        else if(marker==2){
            if(board[1]=='O'||board[1]=='X'){
                    randomnumber=1;
                    printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                    Sleep(2000);
                    showboard(board);
                    continue;
                }
            board[1]='O';
        }
        else if(marker==3){
            if(board[2]=='O'||board[2]=='X'){
                    randomnumber=1;
                    printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                    Sleep(2000);
                    showboard(board);
                    continue;
                }
            board[2]='O';
        }
        else if(marker==4){
            if(board[3]=='O'||board[3]=='X'){
```

Fig. 3.1 b

3.2 GAMEPLAY

Different header C files are included for execution of program. A 1 D array is initialized from digits 1 to 9 for showing positions each time user enters one to make his task easier.

A user defined function ***showboard ()*** displays the array elements in the form of a matrix or a tic tac toe board. The places in 3 x 3 board are the array elements . Our function displays the array elements in form the board.

The game board implementation involves the usage of a simple interface of the original board of TIC-TAC-TOE which has 9 squares, in the form of a square matrix-like denotation, as per the below diagram :

1	2	3
4	5	6
7	8	9

Fig. 3.2 a

This shape of the game board can be easily implemented by using individual parts as multiple lines, vertical and horizontal underlines, forming the board and issuing textbox-like illusions over each of the playing areas, that is square-like boxes numbered from 1-9.

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include<windows.h>

//FUNCTION TO PRINT THE BOARD
void showboard(char board[]){
    printf("\n");
    printf("                                        \n");
    printf("|            |            |           | \n");
    printf("|            |            |           |\n");
    printf("|      %c     |      %c     |      %c    |\n", board[0],board[1],board[2]);
    printf("|            |            |           |\n");
    printf("|____________|____________|___________|\n");
    printf("|            |            |           |\n");
    printf("|            |            |           |\n");
    printf("|      %c     |      %c     |      %c    |\n", board[3],board[4],board[5]);
    printf("|            |            |           |\n");
    printf("|____________|____________|___________|\n");
    printf("|            |            |           |\n");
    printf("|            |            |           |\n");
    printf("|      %c     |      %c     |      %c    |\n", board[6],board[7],board[8]);
    printf("|            |            |           |\n");
    printf("|____________|____________|___________|\n");
}
```

Fig. 3.2 b

3.3 PLAYMAKING

Now, we have the task of filling our tic-tac-toe board alternately. First, each position of our board is associated with our 1 D array element.

We check if first player plays first which will enter its respective sign: ‘X’ or ‘O’, and then we fill the valid position with its respective sign and then change the choice of entered player to other player so now he/she may enter its chance.

We also check if a position does not get filled again, and if the person tries to mark on the position which has already been occupied then the program shows an error prompt and the chance gets shift to the other player, then we print the updated board using the ***showboard()*** function.

```
for(int i=1;i<=18;i++)
{
    if(randomnumber==0)
    {
        printf("\n%s's turn(O): ",oplayer);
        fflush(stdin);
        scanf("%d",&marker);
        system("cls");  //CLEAR THE WHOLE SCREEN
        if(marker==1){
                if(board[0]=='O'||board[0]=='X'){
                    randomnumber=1;
                    printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                    Sleep(2000);   //THE PROGRAM SLEEPS FOR THE SPECIFIED TIME AND CONTINUES FROM THE EXISTING PLACE
                    showboard(board);
                    continue;
                }
                board[0]='O';
        }
```

Fig. 3.3

3.4 RESULT

A game of tic-tac-toe can end in only two possibilities: a winner is set, or both players have come to a draw (tie). In our program we have a user-defined function called ***checkboardfull ( ),*** which tells us whether the board is full or not. The function ***checkboardfull ( )*** returns the integer 1, if the board is not full, otherwise it returns 0 if the board is full. If the board is fully filled and none of the player won the game, then the prompt telling “the game is draw (tie)” is shown in the screen and the program terminates.

The ***checkwhoisthewinner ( )*** function checks if any player has won by checking row wise, column wise or both the diagonals if filled with same . If any one of these is satisfied then it returns the integer 1, and if the above conditions are not fulfilled then it returns 0. Our loop terminates in main () as we get a winner and his name is displayed.

The function ***checkboardfull ( )*** and ***checkwhoisthewinner ( )*** functions are called after every chance is marked on the board so that the correct record of the winner and the status of the board can be maintained.

```
//FUNCTION TO CHECK IF THE BOARD IS FULL OR NOT
int checkboardfull(char board[]){
    int i;
    for(i=0;i<9;i++){
        if(board[0]=='1'||board[1]=='2'||board[2]=='3'||board[3]=='4'||board[4]=='5'||board[5]=='6'||board[6]=='7'||board[7]=='8'||board[8]=='9'){
            return 1;
        }
    }
    return 0;
}
```

Fig. 3.4 a

```c
//FUNCTION TO CHECK WHO IS THE WINNER
int checkwhoisthewinner(char board[]){
    if((board[0]==board[1])&&(board[1]==board[2])){
        return 1;
    }
    else if((board[3]==board[4])&&(board[4]==board[5])){
        return 1;
    }
    else if((board[6]==board[7])&&(board[7]==board[8])){
        return 1;
    }
    else if((board[0]==board[3])&&(board[3]==board[6])){
        return 1;
    }
    else if((board[1]==board[4])&&(board[4]==board[7])){
        return 1;
    }
    else if((board[2]==board[5])&&(board[5]==board[8])){
        return 1;
    }
    else if((board[0]==board[4])&&(board[4]==board[8])){
        return 1;
    }
    else if((board[2]==board[4])&&(board[4]==board[6])){
        return 1;
    }
    else{
        return 0;
    }
}
```

Fig. 3.4 b

3.5 SOME FUNCTIONS

Sleep ( ) function is used in the program, it stops the program for the number of the second specified in its argument.

system("cls") is used to clear the screen fully.

```c
if(randomnumber==0)
{
    printf("\n%s's turn(O): ",oplayer);
    fflush(stdin);
    scanf("%d",&marker);
    system("cls");  //CLEAR THE WHOLE SCREEN
    if(marker==1){
            if(board[0]=='O'||board[0]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                Sleep(2000);   //THE PROGRAM SLEEPS FOR THE SPECIFIED TIME AND CONTINUES FROM THE EXISTING PLACE
                showboard(board);
                continue;
            }
            board[0]='O';
    }
```

Fig. 3.5

3.6 EXECUTION

Fig. 3.6 a

Fig. 3.6 b

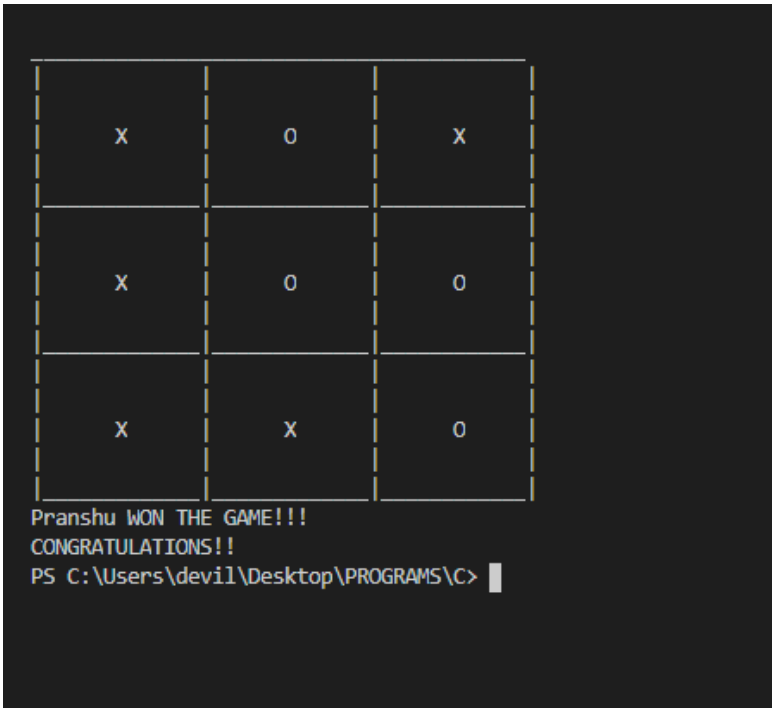
X O X
X O O
X X O
Pranshu WON THE GAME!!!
CONGRATULATIONS!!
PS C:\Users\devil\Desktop\PROGRAMS\C>

Fig. 3.6 c

APPENDIX

Code

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<time.h>
#include<windows.h>

//FUNCTION TO PRINT THE BOARD
void showboard(char board[]){
   printf("\n");
   printf("_____\n");
   printf("|           |           |           | \n");
   printf("|           |           |           |\n");
   printf("|     %c     |     %c     |     %c     |\n", board[0],board[1],board[2]);
   printf("|           |           |           |\n");
   printf("|_____|_____|_____|\n");
   printf("|           |           |           |\n");
   printf("|           |           |           |\n");
   printf("|     %c     |     %c     |     %c     |\n", board[3],board[4],board[5]);
   printf("|           |           |           |\n");
   printf("|_____|_____|_____|\n");
   printf("|           |           |           |\n");
   printf("|           |           |           |\n");
   printf("|     %c     |     %c     |     %c     |\n", board[6],board[7],board[8]);
   printf("|           |           |           |\n");
   printf("|_____|_____|_____|\n");
}

//FUNCTION TO CHECK WHO IS THE WINNER
int checkwhoisthewinner(char board[]){
   if((board[0]==board[1])&&(board[1]==board[2])){
      return 1;
   }
   else if((board[3]==board[4])&&(board[4]==board[5])){
      return 1;
```

```
    }
    else if((board[6]==board[7])&&(board[7]==board[8])){
        return 1;
    }
    else if((board[0]==board[3])&&(board[3]==board[6])){
        return 1;
    }
    else if((board[1]==board[4])&&(board[4]==board[7])){
        return 1;
    }
    else if((board[2]==board[5])&&(board[5]==board[8])){
        return 1;
    }
    else if((board[0]==board[4])&&(board[4]==board[8])){
        return 1;
    }
    else if((board[2]==board[4])&&(board[4]==board[6])){
        return 1;
    }
    else{
        return 0;
    }
}

//FUNCTION TO CHECK IF THE BOARD IS FULL OR NOT
int checkboardfull(char board[]){
    int i;
    for(i=0;i<9;i++){

if(board[0]=='1'||board[1]=='2'||board[2]=='3'||board[3]=='4'||board[4]=='5'||board[5]=='6'||board[6]=='7'||b
oard[7]=='8'||board[8]=='9'){
            return 1;
        }
    }
    return 0;
}

//DRIVER CODE
```

```
void main()
{
   char board[]={'1','2','3','4','5','6','7','8','9'};
   printf("\n\n\n----------WELCOME TO TIC-TAC-TOE----------");
   printf("\n\nEnter the number to mark your chance (X/O):\n\n");
   showboard(board);

   //declaring the string
   char firstplayer[21],xplayer[21],oplayer[21];
   printf("\n\nEnter the nickname of X player: ");
   scanf("%[^\n]s",xplayer);
   fflush(stdin);
   printf("Enter the nickname of O player: ");
   scanf("%[^\n]s",oplayer);

  if((strlen(xplayer)>20)||(strlen(oplayer)>20))
   {
      printf("\nERROR!! Enter a smaller nickname.\n TRY AGAIN!!");
      exit(1);
   }
   int marker,checkwinner=0, boardfull=0, flag=0, randomnumber=0;
   fflush(stdin);
   printf("Enter the name of the player who want to play first:");
   scanf("%[^\n]s",firstplayer);
   if(strcmp(firstplayer,oplayer)==0){
      randomnumber=0;
   }
   else if(strcmp(firstplayer,xplayer)==0){
      randomnumber=1;
   }
   else{
      printf("The player name: %s does not exist.",firstplayer);
      exit(1);
   }

   for(int i=1;i<=18;i++)
   {
```

```
    if(randomnumber==0)
    {
       printf("\n%s's turn(O): ",oplayer);
       fflush(stdin);
       scanf("%d",&marker);
       system("cls");  //CLEAR THE WHOLE SCREEN
       if(marker==1){
            if(board[0]=='O'||board[0]=='X'){
               randomnumber=1;
               printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
               Sleep(2000);   //THE PROGRAM SLEEPS FOR THE SPECIFIED TIME AND
CONTINUES FROM THE EXISTING PLACE
               showboard(board);
               continue;
            }
            board[0]='O';
       }
       else if(marker==2){
          if(board[1]=='O'||board[1]=='X'){
               randomnumber=1;
               printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
               Sleep(2000);
               showboard(board);
               continue;
            }
          board[1]='O';
       }
       else if(marker==3){
          if(board[2]=='O'||board[2]=='X'){
               randomnumber=1;
               printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
               Sleep(2000);
               showboard(board);
               continue;
            }
          board[2]='O';
```

```
        }
        else if(marker==4){
           if(board[3]=='O'||board[3]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
              }
           board[3]='O';
        }
        else if(marker==5){
           if(board[4]=='O'||board[4]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
              }
           board[4]='O';
        }
        else if(marker==6){
           if(board[5]=='O'||board[5]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
              }
           board[5]='O';
        }
        else if(marker==7){
           if(board[6]=='O'||board[6]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
```

```
                Sleep(2000);
                showboard(board);
                continue;
            }
        board[6]='O';
    }
    else if(marker==8){
        if(board[7]=='O'||board[7]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
            }
        board[7]='O';
    }
    else if(marker==9){
        if(board[8]=='O'||board[8]=='X'){
                randomnumber=1;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
            }
        board[8]='O';
    }
    showboard(board);
    checkwinner=checkwhoisthewinner(board);
    if(checkwinner==1){
        printf("%s WON THE GAME!!!\nCONGRATULATIONS!!",oplayer);
        break;
    }
    else{
        boardfull=checkboardfull(board);
        if(boardfull==0){
            printf("GAME TIED!!\nBETTER LUCK NEXT TIME");
```

```
                break;
            }
            else{
                randomnumber=1;
                continue;
            }
        }

    }
    else
    {
        printf("\n%s's turn(X): ",xplayer);
        fflush(stdin);
        scanf("%d",&marker);
        system("cls");  //CLEAR THE WHOLE SCREEN
        if(marker==1){
            if(board[0]=='O'||board[0]=='X'){
                    randomnumber=0;
                    printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                    Sleep(2000);
                    showboard(board);
                    continue;
                }
            board[0]='X';
        }
        else if(marker==2){
            if(board[1]=='O'||board[1]=='X'){
                    randomnumber=0;
                    printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                    Sleep(2000);
                    showboard(board);
                    continue;
                }
            board[1]='X';
        }
        else if(marker==3){
```

```
            if(board[2]=='O'||board[2]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
              }
            board[2]='X';
        }
        else if(marker==4){
            if(board[3]=='O'||board[3]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
              }
            board[3]='X';
        }
        else if(marker==5){
            if(board[4]=='O'||board[4]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
                continue;
              }
            board[4]='X';
        }
        else if(marker==6){
            if(board[5]=='O'||board[5]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will
play.");
                Sleep(2000);
                showboard(board);
```

```
                continue;
            }
        board[5]='X';
    }
    else if(marker==7){
        if(board[6]=='O'||board[6]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                Sleep(2000);
                showboard(board);
                continue;
            }
        board[6]='X';
    }
    else if(marker==8){
        if(board[7]=='O'||board[7]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                Sleep(2000);
                showboard(board);
                continue;
            }
        board[7]='X';
    }
    else if(marker==9){
        if(board[8]=='O'||board[8]=='X'){
                randomnumber=0;
                printf("Space already occupied.\nYou lost your current chance.\nNow the other player will play.");
                Sleep(2000);
                showboard(board);
                continue;
            }
        board[8]='X';
    }
    showboard(board);
```

```
        checkwinner=checkwhoisthewinner(board);
        if(checkwinner==1){
           printf("%s WON THE GAME!!!\nCONGRATULATIONS!!",xplayer);
           break;
        }
        else{
           boardfull=checkboardfull(board);
           if(boardfull==0){
              printf("GAME TIED!!\nBETTER LUCK NEXT TIME");
              break;
           }
           else{
              randomnumber=0;
              continue;
           }
        }
     }
  }
}
```

REFERENCE

1. Griffiths and Griffiths : Head First C, 1st Edition, O'Reilly Publications.

2. Referenced Sleep() in C/C++ on the web at www.softwaretestinghelp.com, given on the link https://www.softwaretestinghelp.com/cpp-sleep/.

3. Referenced about Object-Oriented Programming on the web at www.tutorialspoint.com, as per the hyperlink https://www.tutorialspoint.com/What-is-object-oriented-programming-OOP.

4. Referenced about the rules and structure of the gameplay of the game TIC-TAC-TOE on the web at www.exploratorium.edu, as per the hyperlink reference given as https://www.exploratorium.edu/ brain_explorer/tictactoe.html.

5. Referenced about Implementation of TIC-TAC-TOE game in C on the web at www.geeksforgeeks.com, as per the hyperlink https://www.geeksforgeeks.org/implementation-of-tic-tac-toe-game/ .

6. Referenced about how to better name the functions used in the code on the web at www.medium.org, on the hyperlink https://medium.com/@friskovec.miha/how-to-better-name- your-functions-and-variables-e962a4ef335b.