SCHOOL OF MECHANICAL AND MANUFACTURING ENGINEERING (SMME), NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY (NUST) H-12, ISLAMABAD

PROJECT LAB REPORT

(TIC TAC TOE GAME)

Name: Eemaan Khalid

CMS ID: 464000

Program: BE-Aerospace, Fall 2023

Course code: CS-109 (Fundamentals of Programming-I)

Instructor: Ms. Laiba Waheed

Date: December 29th, 2023

Lab Report

Introduction

Tic Tac Toe is a paper game with a 3 x 3 grid in it. In the game, the two players use noughts and crosses to fill the grid in alternate moves. Whoever makes a row, column, or diagonal with the same symbol wins the game. In programming, the game can be designed to be played either between two users or between the computer and one user.

Project Objective

The objective of the project was to design a program for making a tic-tac-toe game using programming through C++. The program should be designed in a way to deal with as many problems that can occur in the game like double inputting at the same point and providing an invalid input etc.

Project Description

A C++ code for making a tic-tac-toe game employs many basic concepts. It involves the usage of functions, arrays, selection statements, conditional statements, switch cases, and loops. Below here are the further details of each part of the code written for the project:

Code:

The code for the project is given below:

1. Starting with including the libraries and the header files required for the whole code.
   ```
   #include <iostream>
   using namespace std;
   ```

2. Next, we move towards the initialization and declaration of variables used for any data with a specific data type like a 2D order 3 array and initialization of the first turn to X player.
   ```
   char a[3][3] = {{'1', '2', '3'}, {'4', '5', '6'}, {'7', '8', '9'}};
   char turn = 'X'; //defining the first turn
   int r, c; //
   bool make = false;
   ```

3. Further on, display the grid table in which the players will input the desired symbols. Here, a void function is used which shows the symbols assigned for player 1 and player 2 and displays the formation of the grid table using underscores and straight lines. Also, the grid table involves the use of array elements of the initialized array written at its desired places according to the requirement.

```
//function to display the basic table for the game
void showTable()
{
   cout << endl;
   cout << " TIC TAC TOE GAME " << endl;
   cout << " 1st Player : X" << endl;
   cout << " 2nd Player : O" << endl;
   cout << endl;

   cout << "     |     |     " << endl;
   cout << "   " << a[0][0] << "  |  " << a[0][1] << "  |  " << a[0][2] << endl;
   cout << " _____|_____|_____ " << endl;
   cout << "     |     |     " << endl;
   cout << "   " << a[1][0] << "  |  " << a[1][1] << "  |  " << a[1][2] << endl;
   cout << " _____|_____|_____ " << endl;
   cout << "     |     |     " << endl;
   cout << "   " << a[2][0] << "  |  " << a[2][1] << "  |  " << a[2][2] << endl;
   cout << "     |     |     " << endl;
   cout << endl;
}
```

4. In the next step, another function is made to design alternate turning of players as if one player has done his turn, it shifts to the other person's turn. Furthermore, switch cases are used to make choices at different places in the grid. Moreover, there is a checker for the already filled cells.

```
//function for alternate turns of the user
void turnChanging()
{
   int choice;

  if (turn == 'X')
   {
    cout << "Player 1 turn : ";
   }
   if (turn == 'O')
   {
      cout << "Player 2 turn : ";
   }
   cin >> choice;

//using switch so user can select different cells
   switch (choice)
   {
```

```
    case 1: r = 0;  c = 0;  break;
    case 2: r = 0;  c = 1;  break;
    case 3: r = 0;  c = 2;  break;
    case 4: r = 1; c = 0;  break;
    case 5: r = 1; c = 1;  break;
    case 6: r = 1; c = 2;  break;
    case 7: r = 2; c = 0;  break;
    case 8: r = 2; c = 1;  break;
    case 9: r = 2; c = 2;  break;
    default:
    cout << "Your choice is invalid. Please try again." << endl;
    return;
   }
   // Checker for already choosen cells
  if (a[r][c] == 'X' || a[r][c] == 'O')
 {
  cout << "Cell is already filled. Please try again." << endl;
  return;
  }
 //Switching in alternate turns
 a[r][c] = turn;
 if (turn == 'X')
 {
 turn = 'O';
 } else {
            turn = 'X';
 }
 }
```

5. Next, there is a function to check the rows, columns and diagonals to find out if any player wins and there is a checker to check if the game draws when the grid fills without any player winning.

```
    // Function to check if the game is over/ continued or draw
    bool gameOver()
     {
```

```
        char winner = ' ';

// Check rows
for (int i = 0; i < 3; i++)
{
  if (a[i][0] == a[i][1] && a[i][0] == a[i][2])
    winner = a[i][0];
}

// Check columns
for (int i = 0; i < 3; i++)
{
    if (a[0][i] == a[1][i] && a[0][i] == a[2][i])
    winner = a[0][i];
}

// Check diagonals
if (a[0][0] == a[1][1] && a[0][0] == a[2][2])
{
        winner = a[0][0];
        }

if (a[0][2] == a[1][1] && a[0][2] == a[2][0])
{
        winner = a[0][2];
        }
        if (winner != ' ')
{
    cout << "Player " << (winner == 'X' ? "1" : "2") << " wins!" << endl;
    return true;
}

// Check if the board is full (draw)
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        if (a[i][j] != 'X' && a[i][j] != 'O')
            return false;
    }
}
cout << "The game is a draw!" << endl;
return true;
```

```
}
```

6. Next, we use the main function in which we call all the other functions.

```
int main()
{
while (!gameOver())
{
   showTable();
   turnChanging();
   showTable;
}
return 0;
}
```

Output:

The above code provides us with a 3 x 3 grid with numbers from 1 to 9 having three numbers in each row. It also shows which player has its turn. Furthermore, if any player provides an invalid input like he enters two or three-digit number or if he enters the number from the grid that is already filled, the code displays a corresponding output related to the situation. Also, the code also gives the output as which player wins the game as well as tells when the game draws.

Below are the given outputs in each case.

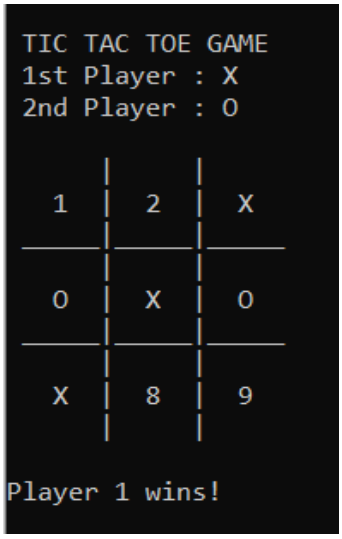

Figure 1(1st case)

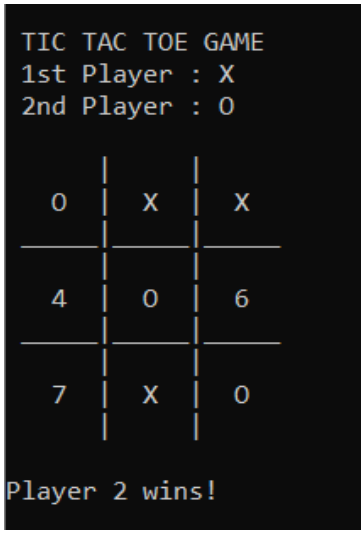

Figure 2(2nd case)

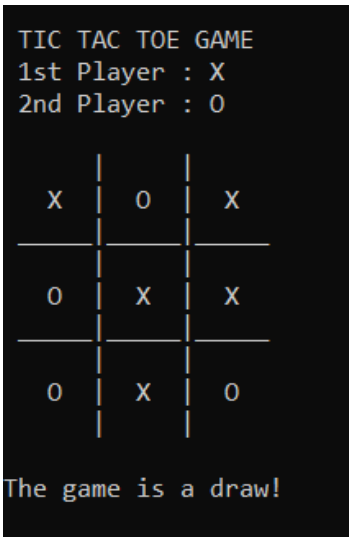

Figure 3(3rd case)