COMPUTER SCIENCE

TIC-TAC-TOE GAME

Done by:

Sparsh Wabhale

CONTENTS

INTRODUCTION:

- Our project name is Tic-Tac-Toe game.
- This game is very popular and is fairly simple by itself. It is a two-player game.
- In this game, there is a board with n x n squares.
- In our game, it is 3 x 3 squares. The goal of Tic-Tac-Toe is to be one of the players to get three same symbols in a row - horizontally, vertically or diagonally on a 3 x 3 grid.
- The game is casually known as X/O.
- Our project has brought this pencil-paper game into a digital screen through programming.

HISTORY:

- Games played on three-in-a-row boards can be traced back to ancient Egypt, where such game boards have been found on roofing tiles dating from around 1300 BCE.
- An early variation of tic-tac-toe was played in the 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 only had 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 which is also played on a simple grid and requires three pieces in a row to finish, and Picaria, a game of the Puebloans.
- The different names of the game are more recent.
- The first print reference to "noughts and crosses", the British name, appeared in 1858, in an issue of Notes and Queries.

STRATEGY TO WIN THIS GAME (FUN FACT):

A player can play a perfect game of tic-tac-toe (to win or at least, draw) if each time it is his turn to play, he chooses the first available move from the following list, as used in Newell and Simon's 1972 tic-tac-toe program.

1. **Win**: If the player has two in a row, they can place a third to get three in a row.
2. **Block**: If the opponent has two in a row, the player must play the third themselves to block the opponent.
3. **Fork**: Create an opportunity where the player has two threats to win (two non-blocked lines of 2).
4. **Centre**: A player marks the centre. (If it is the first move of the game, playing on a corner gives the second player more opportunities to make a mistake and may therefore be the better choice; however, it makes no difference between perfect players.)
5. **Opposite corner**: If the opponent is in the corner, the player plays the opposite corner.
6. **Empty corner**: The player plays in a corner square.
7. **Empty side**: The player plays in a middle square on any of the 4 sides.

SYSTEM REQUIREMENT:

HARDWARE:

1. Processor: Intel(R) Core (TM) i5-8250U CPU @ 1.60GHz, 1801 Mhz, 4 Core(s), 8 Logical Processor(s).

SOFTWARE:

1. DEV C++ was used for coding and compiling.
2. OS Name: Microsoft Windows 10 Pro.

HEADER FILES AND FUNCTIONS: -

1.	#include<iostream>	(For normal input and output statements)
2.	#include<windows.h>	(Used to clear the screen)
3.	#include<conio.h>	(Used to get value from the user)
4.	int win ();	(Checks win condition with 3 matching X or O).
5.	void board_bg ();	(Displays board background.)
6.	void reset_board();	(Resets board background)

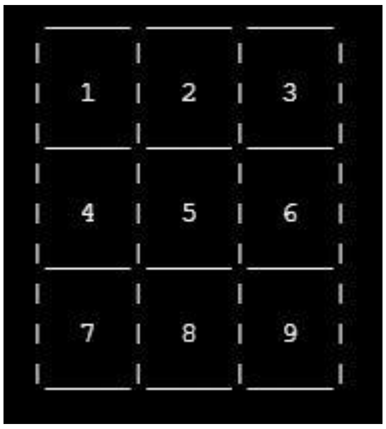
Here important thing to note is that arrays have been used in the code to store the values of X and O and display it on the grid.

PROJECT DESCRIPTION:

How is the program structured?

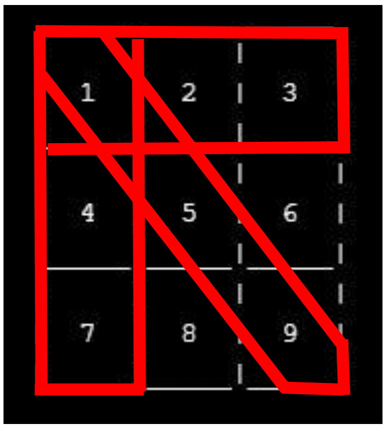
- At the time when program starts, we initialize variables, and we run the game loop until the game end or players choose to quit the game.
  The game consists of three steps:
    - Display board
    - Get players move
    - Check for game end.

The Grid consists of numbered boxes as shown below:



```
 _____ _____ _____
|     |     |     |
|  1  |  2  |  3  |
|_____|_____|_____|
|     |     |     |
|  4  |  5  |  6  |
|_____|_____|_____|
|     |     |     |
|  7  |  8  |  9  |
|_____|_____|_____|
```

- The player enters the numbers through keyboard and it then gets replaced by X or O according to the specified player.
- The player who gets the pattern in horizontal, vertical or diagonal manner wins the game.



- None of the patterns above, leads the game to a draw.
- The game can be continued also without executing the programme again.

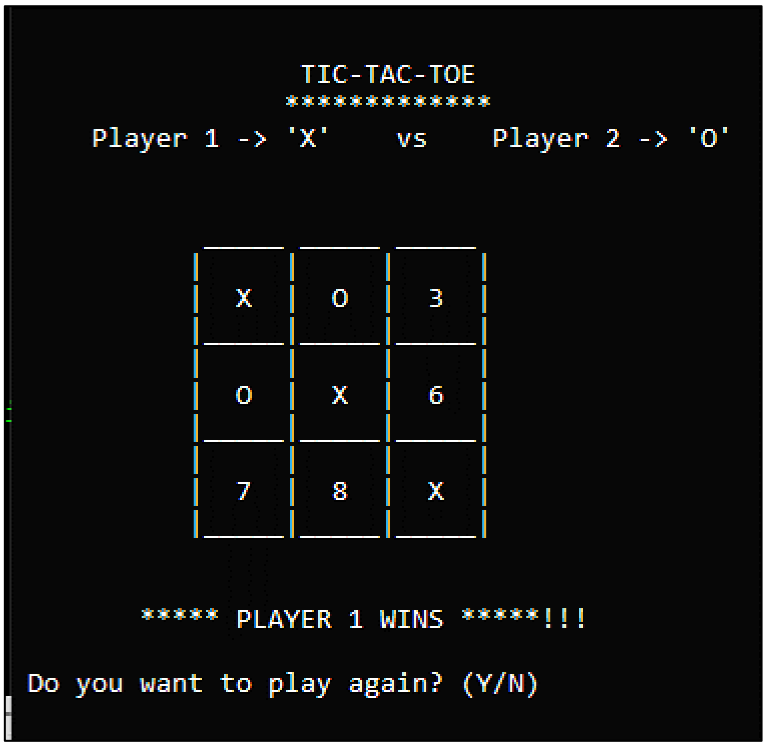
PROGRAM CODE:

```
/*****************COMPUTER SCIENCE INVESTIGATORY PROJECT********************/

Tic-Tac-Toe Game

Done by :- Sparsh Wabhale

SUMMARY:

1. Tic-Tac-Toe Game done using array and functions.

2. Two player game with win and draw conditions.

Version 1.0

/*****************************************************************************/

#include<iostream>
#include<windows.h>
#include<conio.h>
using namespace std;

char square[10] = { '0','1','2','3','4','5','6','7','8','9'} ; // Array to store X and O in board.

                    /****************** FUNCTIONS ********************/

int win();                  // 1. Checks win condition with 3 matching X or O.
void board_bg ();           //  2. Displays board background.
void reset_board();         //  3. Resets board background.

                      /**************************************************/
int player = 1;
int main()
{
      int n,choice;
     char mark , again = 'Y';

do
{
      do
           {
                /******************** SELECTION OF PLAYER********************/

                 system("CLS");
                 board_bg();
                 player = (player % 2)?1:2;                // Selection of player.
                 cout<<"\n Player "<<player<<" Enter a number : ";
                 cin>>choice;
                 mark = (player == 1)? 'X':'O';

                 /**************************************************************/
      /********************INPUT OF 'X' AND 'O' IN THE GRID********************/

                 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;
                 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
                 {
                             cout<<"Invalid move!!!";
                             player--;
                             getch();
                 }
                 n = win();
                 player++;
         }  while(n == -1);

           /***********************************************************/

         board_bg();

         if(n==1)
                 cout<<"\n\n\t\a***** PLAYER "<<--player<<" WINS *****!!!";
         else
                 cout<<"\n\n\t\a GAME DRAW";
         cout<<"\n\n Do you want to play again? (y/n)";
         cin>>again;
         if(again!='y'||again!='Y')
         cout<<"\n\n See you soon.\n Thank you.";

         reset_board();

         }   while(again == 'Y'|| again =='y');
         return 0;
         }
```

```
/******************** TO CHECK IF ANY PAYER HAS WON ********************/

 int win()        // Function to check if a player has won using 3 matching X or O condition.
{
        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[1] == square[5] && square[5] == square[9])
                return 1;
        else if(square[2] == square[5] && square[5] == square[8])
                return 1;
        else if(square[3] == square[6] && square[6] == square[9])
                return 1;
        else if(square[3] == square[5] && square[5] == square[7])
                return 1;
        else if(square[4] == square[5] && square[5] == square[6])
                return 1;
        else if (square[7] == square[8] && square[8] == square[9])
                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;
}
       /*************************************************************/

  /******************** DESIGN AND DISPLAY OF GRID********************/

void board_bg (void)     //Function to display board.
  {
        system("CLS");
        cout<<"\n\n\t\t  Tic-Tac-Toe\n\t\t *************\n";
        cout<<"     Player 1 -> 'X'    vs    Player 2 -> 'O'\n\n\n";
        cout<<"\t    _____ _____ _____ \n";
        cout<<"\t  |     |     |     |\n";
        cout<<"\t  |  "<<square[1]<<"  |"<<"  "<<square[2]<<"  |"<<"  "<<square[3]<<"  |\n";
        cout<<"\t  |_____|_____|_____|\n";
        cout<<"\t  |     |     |     |\n";
        cout<<"\t  |  "<<square[4]<<"  |"<<"  "<<square[5]<<"  |"<<"  "<<square[6]<<"  |\n";
        cout<<"\t  |_____|_____|_____|\n";
        cout<<"\t  |     |     |     |\n";
        cout<<"\t  |  "<<square[7]<<"  |"<<"  "<<square[8]<<"  |"<<"  "<<square[9]<<"  |\n";
        cout<<"\t  |_____|_____|_____|\n";
  }
```

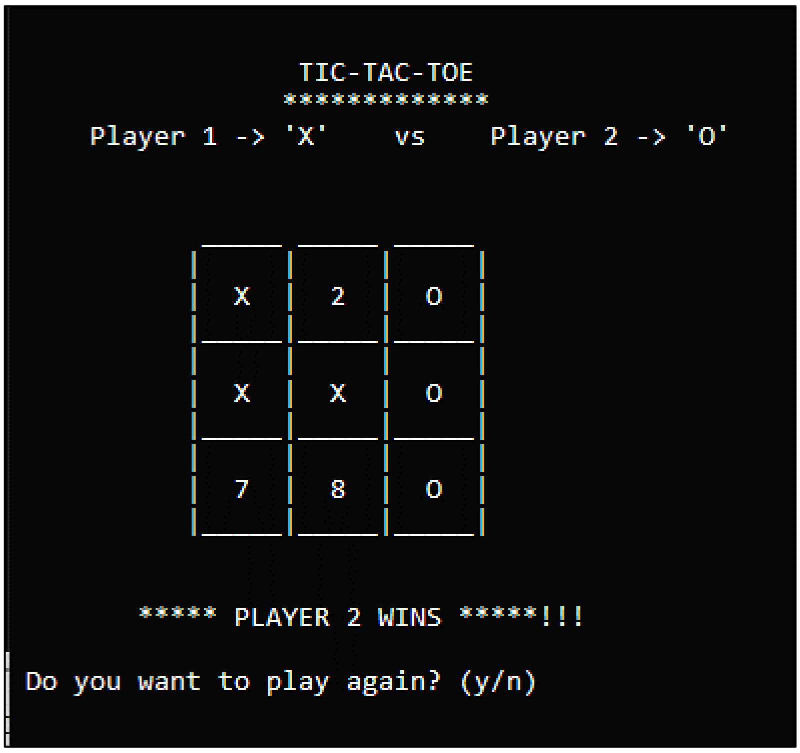
```
/*********************RESET THE GRID FOR NEW GAME*********************/

        void reset_board()    // Function to reset board.
            {
                square[0] = '0';
                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';
                player = 1;
            }

      /***********************END OF THE PROGRAM/CODE********************/
```

OUTPUT:
1) PLAYER 1 WINS!!!!!



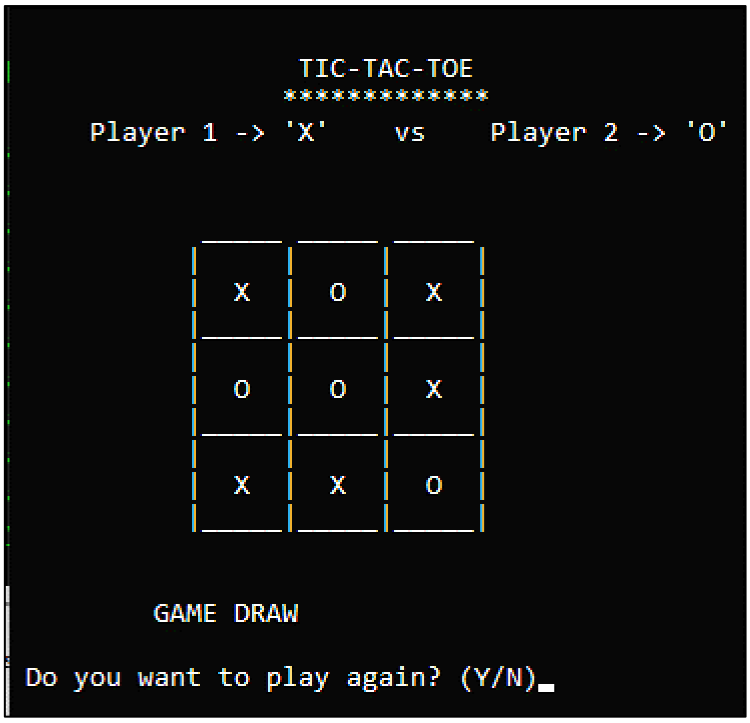
```
                TIC-TAC-TOE
               *************
    Player 1 -> 'X'    vs    Player 2 -> 'O'


             _____ _____ _____
            |     |     |     |
            |  X  |  O  |  3  |
            |_____|_____|_____|
            |     |     |     |
            |  O  |  X  |  6  |
            |_____|_____|_____|
            |     |     |     |
            |  7  |  8  |  X  |
            |_____|_____|_____|


        ***** PLAYER 1 WINS *****!!!

 Do you want to play again? (Y/N)
```

2) PLAYER - 2 WINS!!!!!



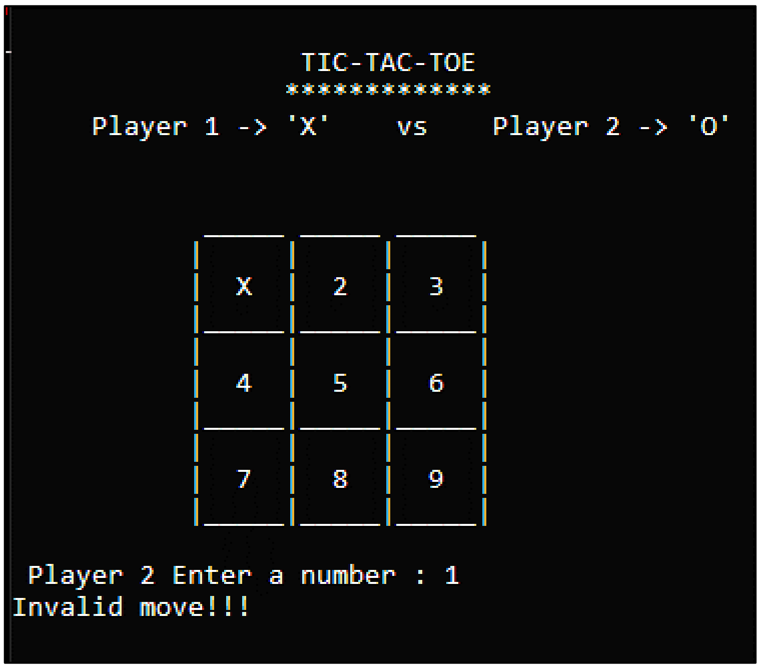
```
               TIC-TAC-TOE
              **************
    Player 1 -> 'X'    vs    Player 2 -> 'O'


             _____ _____ _____
            |     |     |     |
            |  X  |  2  |  O  |
            |_____|_____|_____|
            |     |     |     |
            |  X  |  X  |  O  |
            |_____|_____|_____|
            |     |     |     |
            |  7  |  8  |  O  |
            |_____|_____|_____|


        ***** PLAYER 2 WINS *****!!!

Do you want to play again? (y/n)
```

3) GAME DRAW:



```
               TIC-TAC-TOE
              **************
    Player 1 -> 'X'    vs    Player 2 -> 'O'


             _____ _____ _____
            |     |     |     |
            |  X  |  O  |  X  |
            |_____|_____|_____|
            |     |     |     |
            |  O  |  O  |  X  |
            |_____|_____|_____|
            |     |     |     |
            |  X  |  X  |  O  |
            |_____|_____|_____|


         GAME DRAW

Do you want to play again? (Y/N)_
```

4) INVALID MOVE:



```
                  TIC-TAC-TOE
                 **************
     Player 1 -> 'X'    vs    Player 2 -> 'O'


             _____ _____ _____
            |     |     |     |
            |  X  |  2  |  3  |
            |_____|_____|_____|
            |     |     |     |
            |  4  |  5  |  6  |
            |_____|_____|_____|
            |     |     |     |
            |  7  |  8  |  9  |
            |_____|_____|_____|

 Player 2 Enter a number : 1
Invalid move!!!
```

FLOW OF THE PROGRAM:

1) The numbered grid is displayed with indication of player 1 and 2.
2) The numbers entered by the player is stored and replaced by X or O respectively.
3) If any player gets a straight pattern, the player wins, and it is displayed.
4) If none of the players get a straight line, the game is draw and it is displayed.
5) If any key other than number >9 or alphabets, displays as wrong move.

LIMITATIONS:

1) Making one player as computer – random playing with single player
2) Mouse is not used for input. Only keyboard is used.

BIBLIOGRAPHY:

www.google.com
www.slideshare.com
www.onlinecompiler.in