PROJECT WORK OF COMPUTER PROGRAMMING

TIC TAC TOE

Student Name: Aastha singh	**UID:24BCA10483**
Branch: BCA	**Section/Group: 7-A**
Semester: 1	**Date of performance:**
Subject Name: Computer Programming	**Subject Code:24CAH101**

1. **Aim/overview of the practical:**
   . To develop TIC TAC TOE in the c programming language.

2.Task to be done:

- **Set up and initialize the board.**
- **Display the board** after each move.
- **Take player input** and ensure valid moves.
- **Check for a win or draw** after each move.
- **Run a game loop** until someone wins or the game draws.
- **Ask for a replay** or exit after the game finishes.

3.Algorithm /Flowchart:

```
1. Start
2. Initialize the Game Board: Set up a 3x3 grid with numbers 1 to 9 representing each
   cell.
3. Display the Game Board: Print the current state of the grid.
4. Set Player 1 as 'X' and Player 2 as 'O': Assign markers to the players.
5. Loop until there is a winner or a draw:
    1. Player Turn: Alternate between Player 1 and Player 2.
    2. Prompt Player for Input: Ask the player to choose a position (1-9).
    3. Check for Valid Move: Ensure the chosen position is empty.
    4. Update the Board: Place the player's marker on the chosen position.
    5. Display the Updated Board: Show the new state of the grid.
    6. Check for Win:
        ▪ Check all rows, columns, and diagonals to see if either player has three
          matching markers in a row.
    7. Check for Draw:
        ▪ If all positions are filled and no player has won, declare a draw.
6. End the Game:
```

    - If a player wins, declare that player as the winner.
    - If the game ends in a draw, announce a draw.
7. **Prompt for Replay**:
    - Ask if the players want to play again.
    - If yes, reinitialize the game; if no, exit.
8. **End**

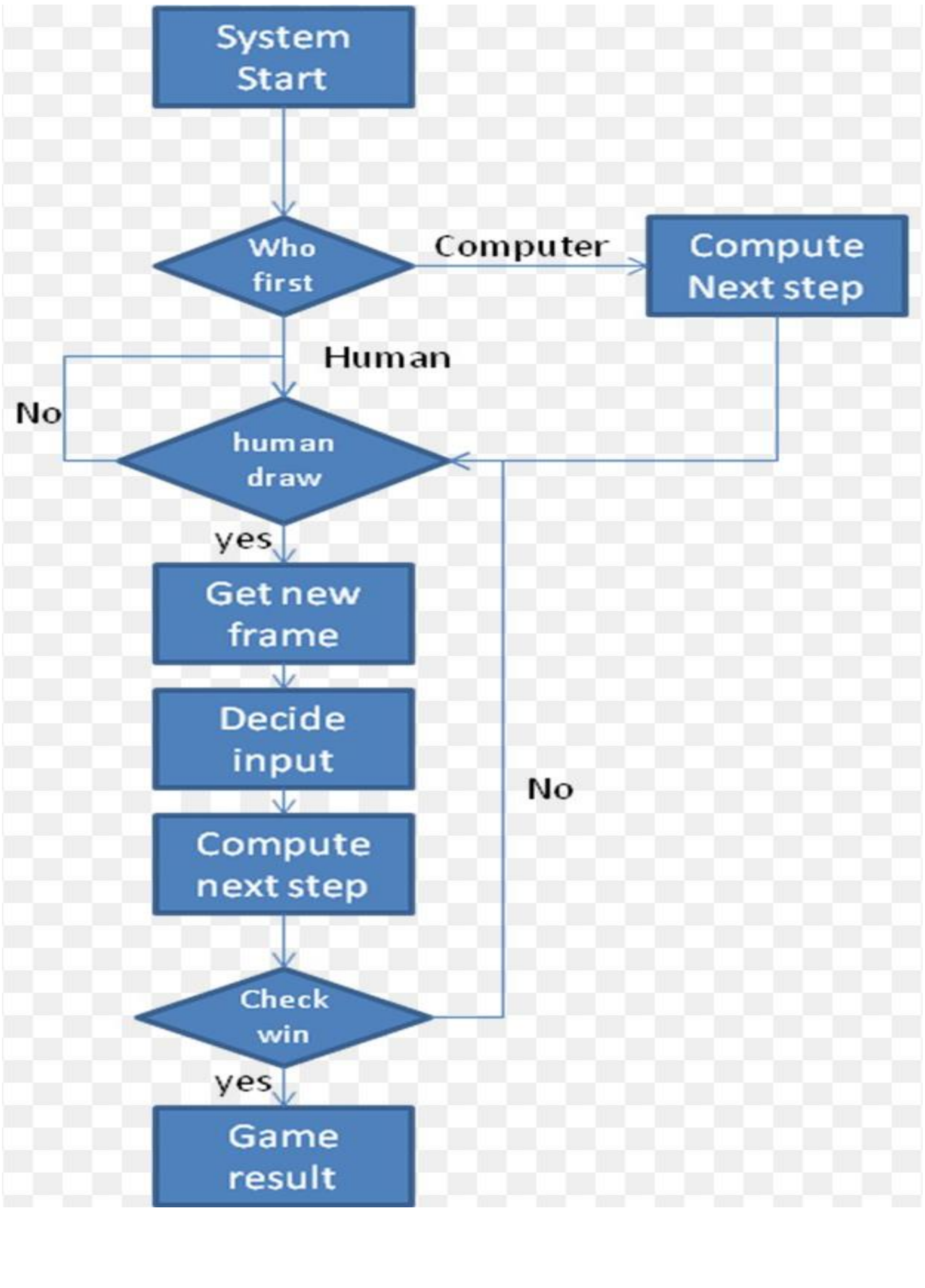
Flowchart of Tic-Tac-Toe:

• **Start**: An oval to represent the start of the program

• **Initialize Board**: A process box to initialize the grid.

• **Display Board**: A process box to print the current state of the board.

• **Player Turn**: A decision diamond to determine whose turn it is.
• **Input Validation**: A decision diamond to check if the input is valid.
• **Check Win/Draw**: Two decision diamonds to check for a win or draw.
•**End**: An oval to represent the end of the program.

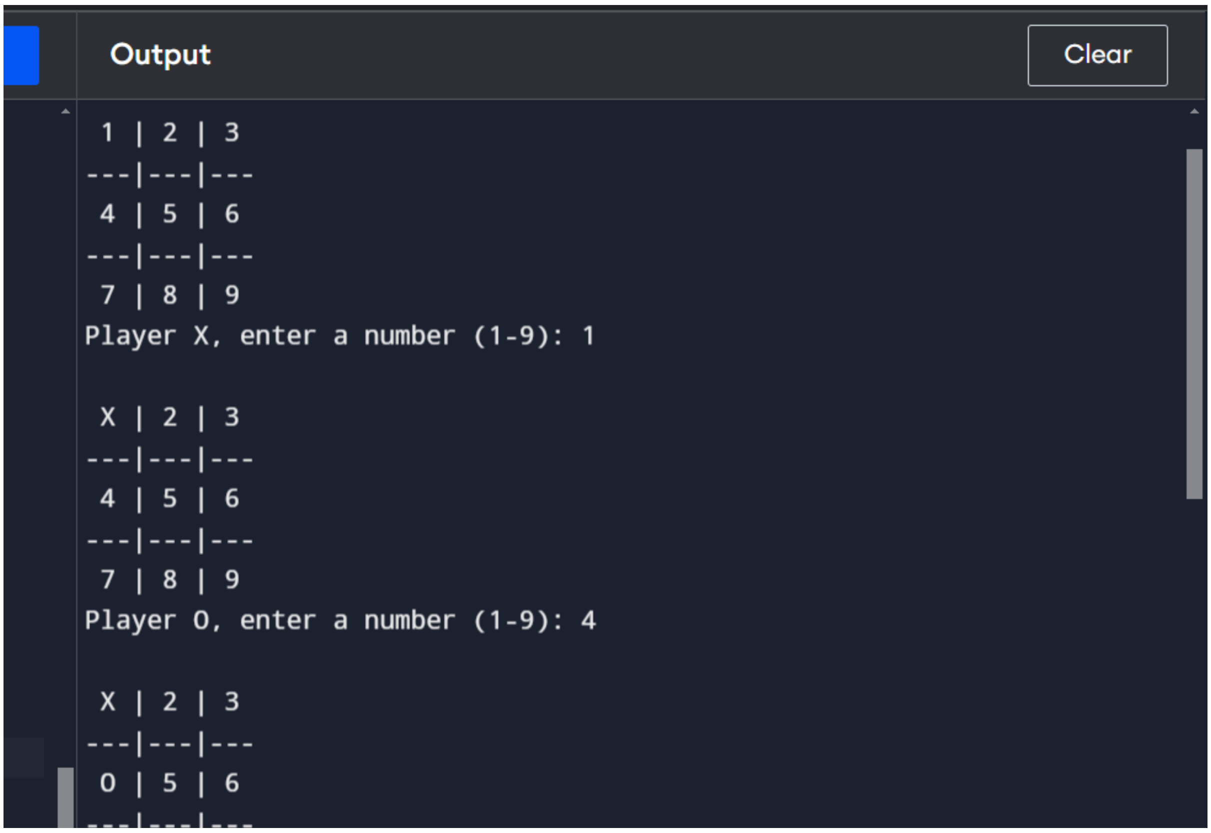

System Start
Who first
Computer
Compute Next step
Human
No
human draw
yes
Get new frame
Decide input
Compute next step
No
Check win
yes
Game result

4. Code for experiment /practical:

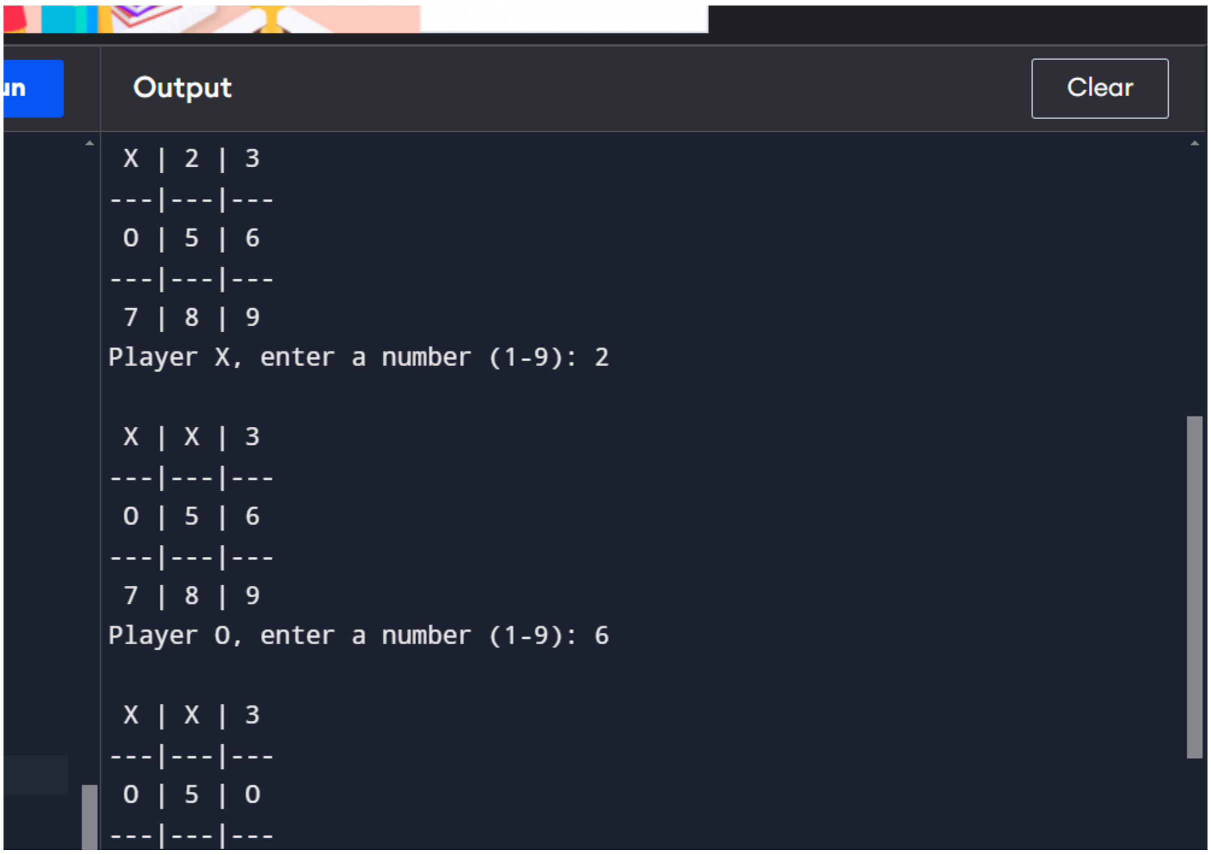
```
#include <stdio.h>

// Global board variable
char board[3][3];

// Function to initialize the board with numbers 1-9
void initialize Board () {
    int count = 1;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            board[i][j] = count + '0'; // Store as characters '1' to '9'
            count++;
        }
    }
}

// Function to display the board
void display Board () {
    printf("\n");
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf(" %c ", board[i][j]);
            if (j < 2) printf("|"); // Vertical lines
        }
        if (i < 2) printf("\n---|---|---\n"); // Horizontal lines
    }
    printf("\n");
}
```

```
// Function to check for a win
int checkWin() {
    // Check rows and columns
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2]) return 1;
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i]) return 1;
    }
    // Check diagonals
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2]) return 1;
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0]) return 1;
    return 0;
}

// Function to check if the game is a draw
int checkDraw() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] != 'X' && board[i][j] != 'O') return 0;
        }
    }
    return 1;
}

// Function to take player's input
void playerMove(char player) {
    int choice;
    int row, col;
    printf("Player %c, enter a number (1-9): ", player);
    scanf("%d", &choice);

    // Convert choice to row and column
    row = (choice - 1) / 3;
    col = (choice - 1) % 3;

    // Check if the chosen cell is empty
    if (board[row][col] != 'X' && board[row][col] != 'O') {
        board[row][col] = player; // Place the player's marker
    } else {
        printf("Invalid move! Try again.\n");
        playerMove(player); // Recursion for invalid input
    }
}

int main() {
    char currentPlayer = 'X'; // Player 1 starts with 'X'
    int gameStatus = 0; // 0: ongoing, 1: win, -1: draw

    initializeBoard(); // Setup initial board

    // Main game loop
    while (1) {
        displayBoard(); // Show current board
        playerMove(currentPlayer); // Get player's move

        // Check if there's a winner
        if (checkWin()) {
            displayBoard(); // Display final state of the board
            printf("Player %c wins!\n", currentPlayer);
            break;
        }

        // Check for a draw
        if (checkDraw()) {
            displayBoard(); // Display final state of the board
            printf("It's a draw!\n");
            break;
        }

        // Switch to the other player
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }

    return 0;
}
```

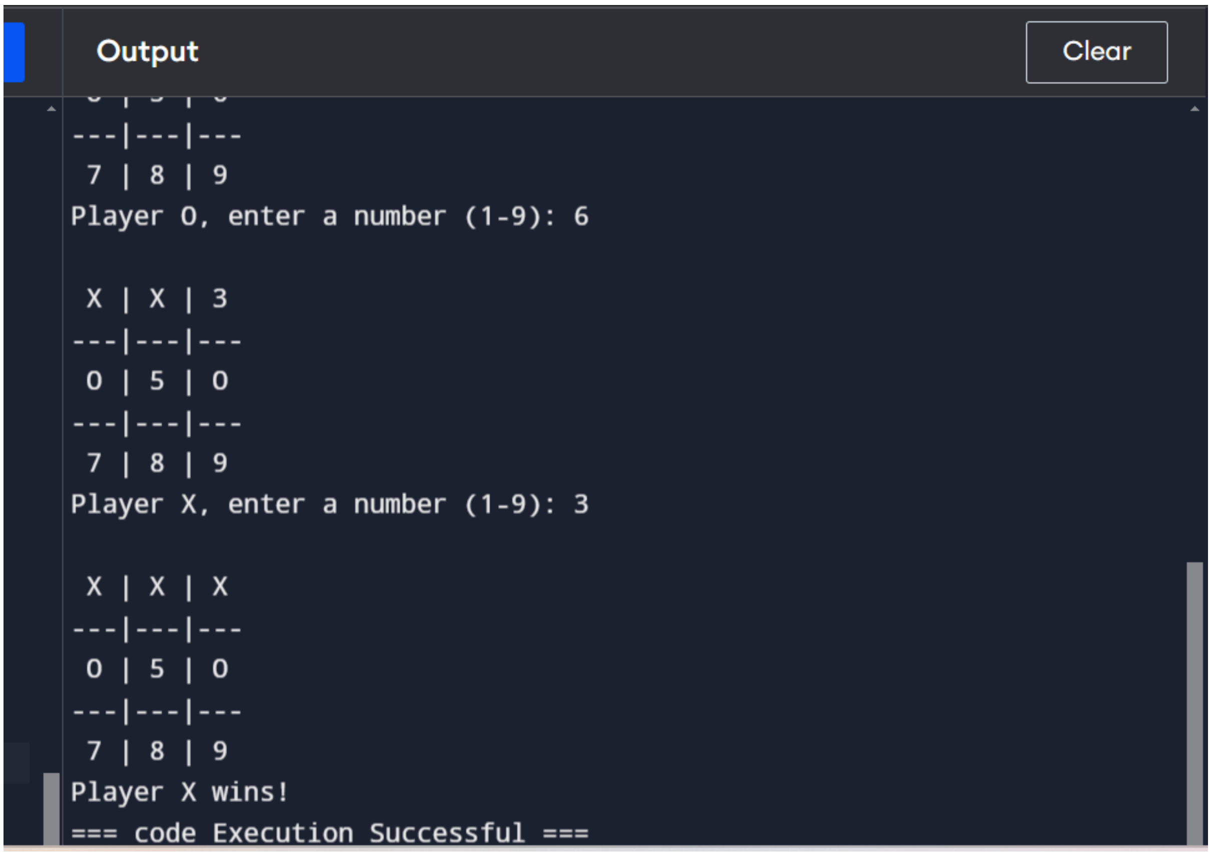
5. Result/Output:



```
 1 | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9
Player X, enter a number (1-9): 1

 X | 2 | 3
---|---|---
 4 | 5 | 6
---|---|---
 7 | 8 | 9
Player O, enter a number (1-9): 4

 X | 2 | 3
---|---|---
 O | 5 | 6
```



```
 X | 2 | 3
---|---|---
 O | 5 | 6
---|---|---
 7 | 8 | 9
Player X, enter a number (1-9): 2

 X | X | 3
---|---|---
 O | 5 | 6
---|---|---
 7 | 8 | 9
Player O, enter a number (1-9): 6

 X | X | 3
---|---|---
 O | 5 | O
---|---|---
```



```
---|---|---
 7 | 8 | 9
Player O, enter a number (1-9): 6

 X | X | 3
---|---|---
 O | 5 | O
---|---|---
 7 | 8 | 9
Player X, enter a number (1-9): 3

 X | X | X
---|---|---
 O | 5 | O
---|---|---
 7 | 8 | 9
Player X wins!
=== code Execution Successful ===
```

6.Writting Summary:

The Tic-Tac-Toe project implemented in C is a simple, interactive two-player game designed to run in a console environment. It enables two players to alternate turns in marking the cells of a 3x3 grid with their respective symbols ('X' for Player 1 and 'O' for Player 2). The primary objective of the game is for a player to get three of their marks in a row—either horizontally, vertically, or diagonally—before their opponent. If neither player manages to do so, and the grid is completely filled, the game results in a draw.

7.Learning Outcomes:

1.Understanding 2D Arrays

2. Input Validation and Error Handling

3. Control Flow and Conditional Logic

4. Game Logic Implementation

5. Building an Interactive Console Application

Evaluation Grid:

SN.NO.	**Parameters**	**Marks Obtained**	**Maximum Marks**
1.	**Demonstration and performance (pre lab Quiz)**		**5**
2.	**Worksheet**		**10**
3.	**Post Lab Quiz**		**5**