TIC-TAC-TOE GAME

Submitted by:

MD. ADNAN ZAHAN ROMEO 242-35-215

ABDUR RAHMAN TANVIR 242-35-736

MD Robayet Hossain Pranto 242-35-490

1. INTRODUCTION

The Tic-Tac-Toe Game is a simple yet engaging two-player game where players take turns marking spaces in a 3x3 grid. The goal is to align three of their respective marks ('X' or 'O') in a row, column, or diagonal before the opponent does. The game is implemented using C programming and runs in a console-based interface.

2. OBJECTIVE

The primary objective of this project is to develop a functional Tic-Tac-Toe game that:

- Allows two players to alternately place their marks.
- Ensures valid moves and prevents overwriting existing marks.
- Determines and announces a winner or a draw.

3. SYSTEM REQUIREMENTS

- **Hardware Requirements:**
    - Processor: Intel Core i3 or higher
    - RAM: 2GB or higher
    - Storage: Minimal (less than 1MB)
- **Software Requirements:**
    - Operating System: Windows, Linux, or macOS
    - Compiler: GCC or any C compiler

- IDE: Code::Blocks, Dev-C++, or Visual Studio Code

4. SYSTEM DESIGN

The system follows a simple procedural programming approach using the C language. The game logic is divided into the following functional modules:

- **initializeBoard()**: Initializes the game board with empty spaces.
- **displayBoard()**: Displays the current state of the board.
- **checkWinner()**: Checks if a player has won or if the game is a draw.
- **makeMove()**: Takes user input and validates the move.
- **main()**: Controls the game flow and execution.

5. IMPLEMENTATION

The program follows a loop where players take turns making moves until either a player wins or the game results in a draw. The game board updates after each turn, and at the end, the winner is announced, or the game is declared a draw if all cells are filled.

6. FLOWCHART

A flowchart representing the execution flow of the game includes:

1. Start
2. Initialize board
3. Display board
4. Take player input
5. Validate move
6. Check for a winner or draw
7. Switch turn
8. Repeat until a result is obtained
9. Display result
10. End

7. TESTING AND OUTPUT

The game was tested with different scenarios, including:

- A player winning by filling a row, column, or diagonal.
- A draw scenario where all cells are filled without a winner.
- Handling of invalid inputs.

8. CONCLUSION

The Tic-Tac-Toe game successfully implements the fundamental mechanics of the classic game using C programming. It provides an interactive experience with a fair and simple design. Future enhancements may include an AI opponent, a graphical interface, or network-based multiplayer functionality.

9. FUTURE ENHANCEMENTS

- Implement AI using the Minimax algorithm.
- Add a graphical user interface (GUI) using graphics libraries.
- Develop an online multiplayer version.

MAIN CODE:

```
#include <stdio.h>


char board[3][3];

int playerTurn = 1;


void initializeBoard() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            board[i][j] = ' ';
        }
    }
```

```
}

void displayBoard() {
    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("|");
        }
        printf("\n");
        if (i < 2) printf("---|---|---\n");
    }
    printf("\n");
}

int checkWinner() {
    for (int i = 0; i < 3; i++) {
        if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != ' ')
            return playerTurn;
        if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != ' ')
            return playerTurn;
    }
    if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != ' ')
        return playerTurn;
    if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != ' ')
        return playerTurn;
```

```
    return 0;
}

void makeMove() {
    int row, col;
    printf("Player %d, enter your move (row and column): ", playerTurn);
    scanf("%d %d", &row, &col);
    if (row < 1 || row > 3 || col < 1 || col > 3 || board[row - 1][col - 1] != ' ') {
        printf("Invalid move! Try again.\n");
        makeMove();
    } else {
        board[row - 1][col - 1] = (playerTurn == 1) ? 'X' : 'O';
    }
}

int main() {
    initializeBoard();
    int winner = 0;
    int moves = 0;

    while (moves < 9 && !winner) {
        displayBoard();
        makeMove();
        moves++;
        winner = checkWinner();
        if (!winner) playerTurn = (playerTurn == 1) ? 2 : 1;
```

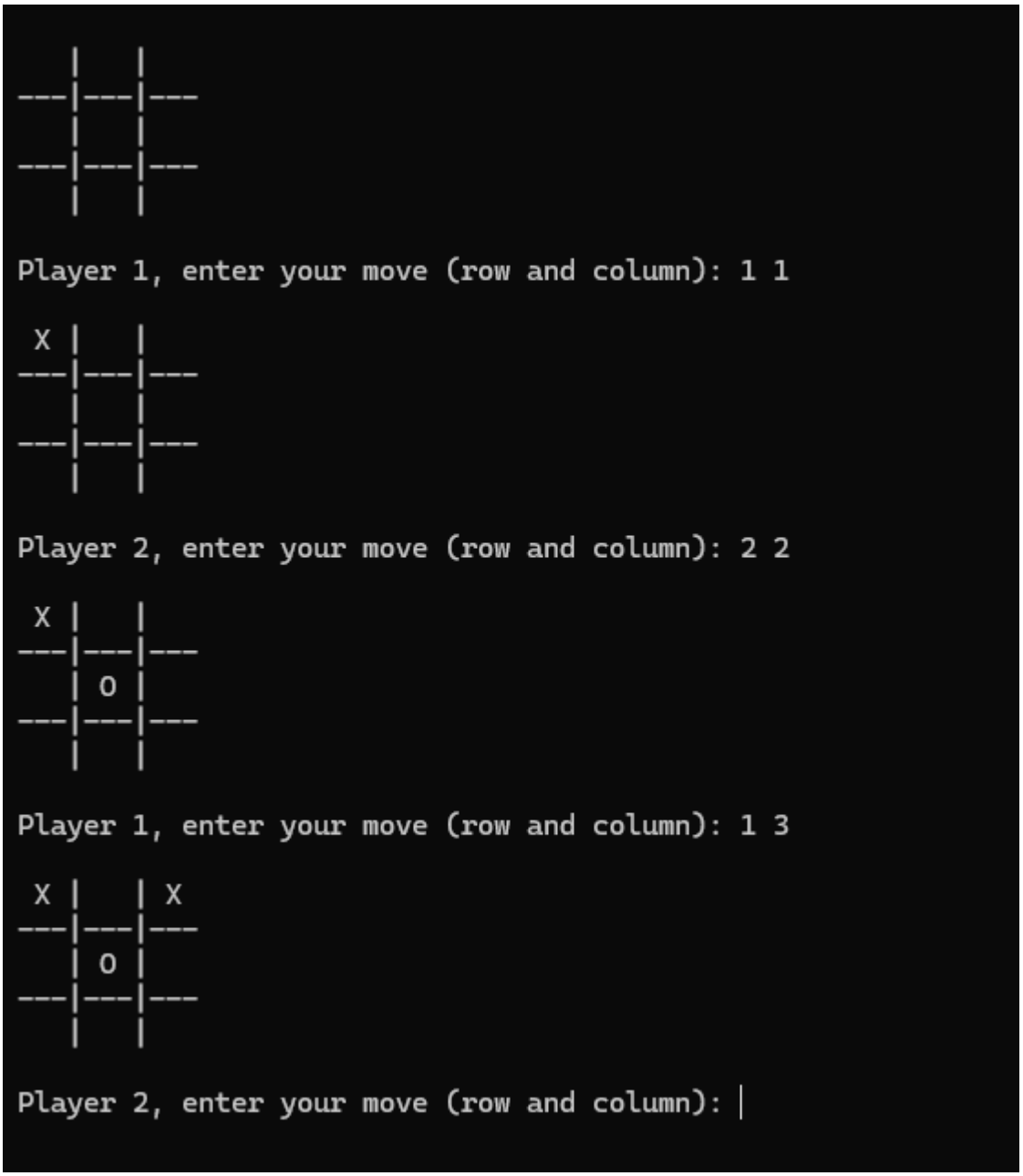
```
    }

    displayBoard();

    if (winner) {
        printf("Player %d wins!\n", winner);
    } else {
        printf("It's a draw!\n");
    }

    return 0;
}
```

OUTPUT :



REFERENCES

- C Programming Documentation
- Online resources and coding tutorials on game development in C.