1. TITLE PAGE

TIC-TAC-TOE GAME USING C (MODULAR PROGRAMMING)

Submitted in partial fulfilment of the requirements for the course
C Programming Project

Student Name: Abhishek Singh
UID: 590027542
Course: B tech CSE
Department: School Of Computer Science
Institution: University OF Petroleum And Energy Studies
Year: 2025

Submitted To: Dr. Srinivasan Ramchandran

2. ABSTRACT

This project implements a console-based Tic Tac-Toe game using the C programming language following modular programming principles. The program supports two modes: Player vs Computer and Player vs Player.

A 3×3 game board is displayed in the terminal, and the program takes user inputs for row and column positions. Input validation ensures that invalid or already-occupied positions are not accepted. The computer player selects its moves using a random number generation method.

The project demonstrates core programming concepts such as arrays, functions, conditional statements, loops, modular design using multiple files, and basic input handling.

---

3. PROBLEM DEFINITION

Tic-Tac-Toe is a simple game played on a 3×3 grid between two players. The challenge is to implement this game using C language while maintaining modular programming structure.

Objectives:

- Implement a working Tic-Tac-Toe game.

- Allow both **Human vs Human** and **Human vs Computer** modes.
- Implement proper input validation and win detection.
- Organize the code into multiple modules using .c and .h files.
- Follow the project folder structure provided by the institution.

---

4. SYSTEM DESIGN

4.1 Flowchart

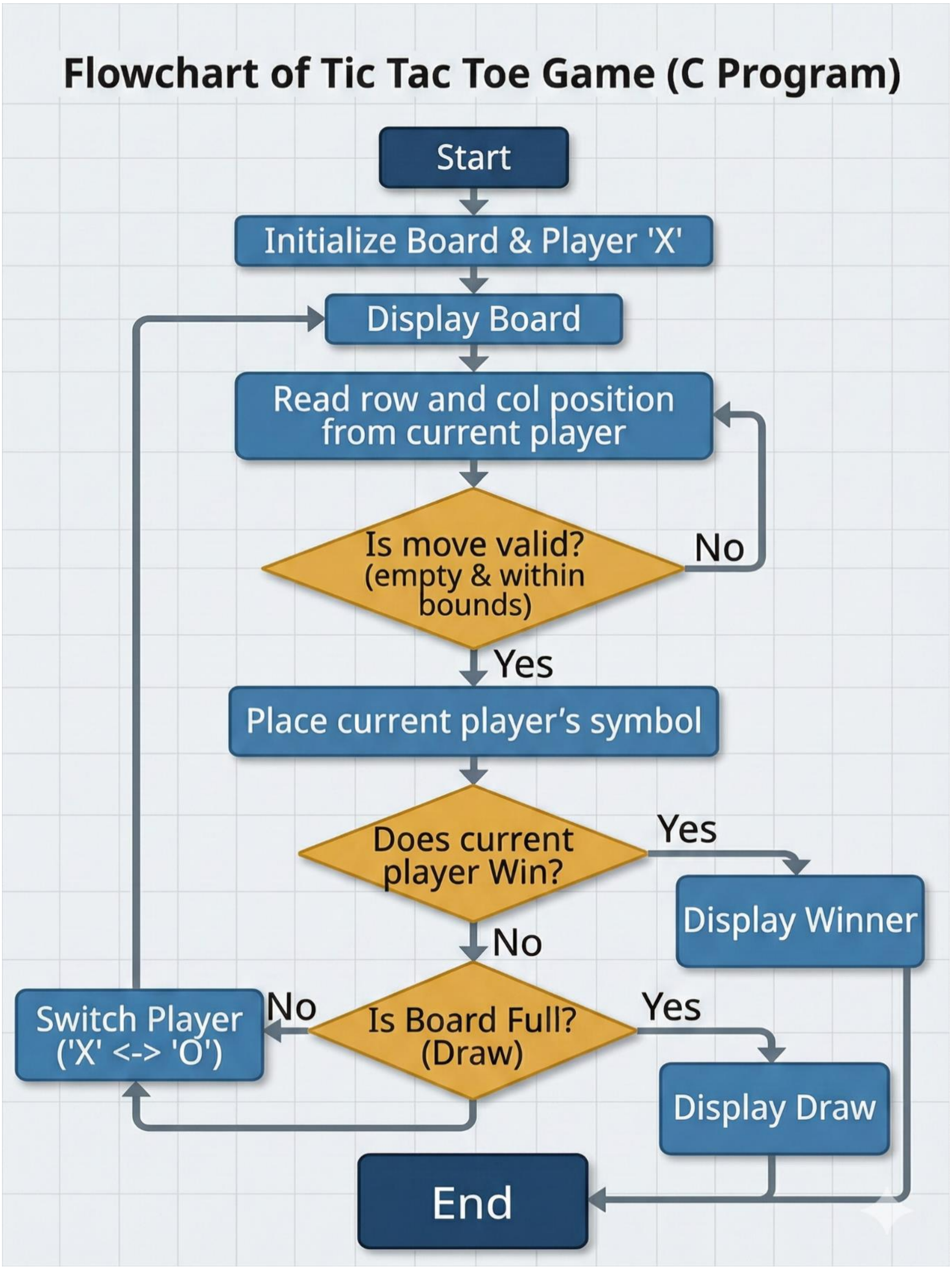

Flowchart of Tic Tac Toe Game (C Program)
Start
Initialize Board & Player 'X'
Display Board
Read row and col position from current player
Is move valid? (empty & within bounds)
No
Yes
Place current player's symbol
Does current player Win?
Yes
Display Winner
No
Is Board Full? (Draw)
No
Switch Player ('X' <-> 'O')
Yes
Display Draw
End

Figure 1: Flowchart of Tic-Tac-Toe Game Logic

4.2 Algorithm

Main Algorithm:

```
1. Display menu
2. Take user mode choice
3. Take player symbol (X or O)
4. Initialize board
5. While game not finished:
    a. Take player input
    b. Validate input
    c. Update board
    d. Check for win or draw
    e. If vs computer:
        Generate random computer move
6. Display winner or draw
7. Exit program
```

---

5. IMPLEMENTATION DETAILS

5.1 Modular File Structure

```
/src
  main.c
  game.c
/include
  tictactoe.h
/docs
/assets
README.md
sample_input.txt
```

5.2 Key Code Snippets

(i) Board Declaration

```
char a[3][3];
```

(ii) Board Initialization

```
void initBoard(void) {
    for(int i = 0; i < 3; i++) {
        for(int j = 0; j < 3; j++) {
            a[i][j] = ' ';    // Initializes the 3x3 board with empty spaces
        }
    }
}
```

(iii) Winner Checking Logic

```
char check(void) {
    for(int i=0; i<3; i++) {
        if(a[i][0]==a[i][1] && a[i][1]==a[i][2] && a[i][0] != ' ')
            return a[i][0];
    }

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

    if(a[0][0]==a[1][1] && a[1][1]==a[2][2] && a[0][0] != ' ')
```

```
        return a[0][0];

    if(a[0][2]==a[1][1] && a[1][1]==a[2][0] && a[0][2] != ' ')
        return a[0][2];

    return ' ';
}
```

(iv) Computer Move Logic

```
void computerRandomMove(char cpu, int *moves) {
    int r, c;
    while(1) {
        r = rand() % 3;
        c = rand() % 3;
        if(a[r][c] == ' ') {
            a[r][c] = cpu;
            (*moves)++;
            break;
        }
    }
}
```

---

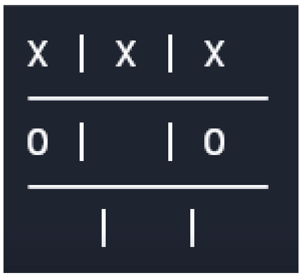
6. TESTING & RESULTS

6. TESTING & RESULTS

To verify the correctness and robustness of the Tic-Tac-Toe game, several test cases were executed covering win conditions, draw scenarios, and invalid inputs. The results were consistent with expected behavior.

Test Case 1: Player Wins (Top Row)

- Input: Player selects symbol X and makes moves at (0,0), (0,1), (0,2)
- Expected Output: You Won!
- Observed Output: You Won!

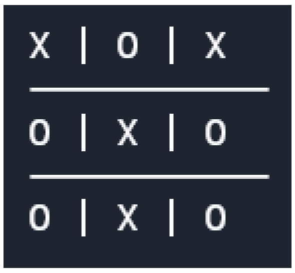


```
X | X | X
---------
O |   | O
---------
  |   |
```

- ✅ Player wins — top row filled with X .

Test Case 2: Draw Match

- Input: All cells filled with no winning line
- Expected Output: Draw Match
- Observed Output: Draw Match

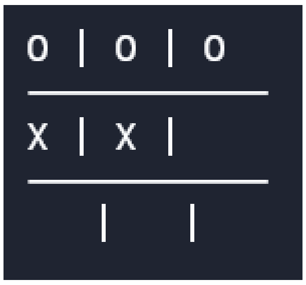


```
X | O | X
---------
O | X | O
---------
O | X | O
```

- ✅ No winner — game ends in a draw.

Test Case 3: Computer Wins

- Input: Player is X , computer is O . Computer fills top row.
- Expected Output: Computer Won.
- Observed Output: Computer Won.



```
O | O | O
---------
X | X |
---------
  |   |
```

- ✅ Computer wins — top row filled with O .

Test Case 4: Invalid Input Handling

- Input: Player enters coordinates outside the board (e.g., 3 3)
- Expected Output: Invalid move. Try again.
- Observed Output: Invalid move. Try again.

```
"Invalid move. Try again."
```

✅ Program rejects invalid input and prompts again.

---

7. CONCLUSION & FUTURE WORK

Conclusion:

The Tic-Tac-Toe game was successfully implemented using modular programming in C. The program works correctly in both game modes and follows the required directory structure.

Future Improvements:

- Smarter AI using strategy algorithms
- GUI-based version
- Scoreboard system
- Save game functionality

---

8. REFERENCES

1. K.N. King, *C Programming – A Modern Approach*
2. GeeksforGeeks – Tic Tac Toe in C
3. YouTube videos