**Tic Tac Toe Game**

**Abstract**

The Tic Tac Toe Game project is a console-based implementation of the traditional Tic Tac Toe game, developed using the C programming language. It incorporates modern programming practices and features that provide an engaging user experience. The game supports two modes:

1. **Single-player mode**, where the user competes against an AI-controlled computer opponent.
2. **Multiplayer mode**, where two players compete against each other.

This project implements robust game mechanics, including error handling for invalid inputs, a dynamic and visually appealing game board, and functionalities to save and load player statistics across game sessions. Additional enhancements include console colorization to distinguish between elements such as player moves, instructions, and statistics. The primary focus is to create a user-friendly and functional gaming experience while demonstrating the principles of modular programming, file handling, and random number generation.

Key features include:

* A 3x3 game board with real-time updates.
* Computer AI capable of making valid random moves.
* A player statistics system to track total games, wins, losses, and draws.
* Persistent storage of statistics via binary file operations.

**Implementation Details**

**a. Functional Overview**

* **Main Menu:** Allows users to start a new game or exit.
* **Game Modes:**
  + **Player vs. Computer:** Compete against a basic AI that place moves randomly.
  + **Player vs. Player:** Two human players take turns on the same system.
* **Game Board Management:** The board is dynamically updated after every move and displayed in a grid format.
* **Win Conditions:** The program checks for three-in-a-row horizontally, vertically, or diagonally to determine a winner.
* **Draw Detection:** If all cells are filled without a winner, the game declares a draw.
* **Statistics Tracking:** Tracks the number of games played, wins, losses, and draws for the user.
* **File Persistence:** Saves player statistics to a binary file (game\_stats.txt) and reloads them during subsequent sessions.

**b. Technical Highlights**

* **Colorized Output:** Enhances readability by using color codes for instructions, the grid, and player turns.
* **Randomized Computer Moves:** Ensures the computer chooses valid cells for its moves.
* **Dynamic Board Initialization:** Resets the grid at the start of every new game.
* **Modular Design:** Functions are organized logically to separate concerns (e.g., checkWin, initializeBoard, computerMove).
* **File I/O:** Implements fwrite and fread for storing and retrieving statistics.

**c. Challenges Overcome**

* Handling invalid inputs gracefully to prevent crashes.
* Avoiding overwriting saved statistics during file operations.
* Ensuring the game remains user-friendly with clear instructions and visual cues.

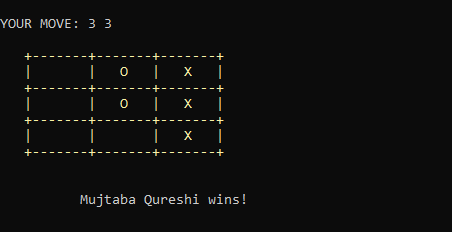
**Welcome Screen and Game Instructions and also the last game statistics**



**Game Grid Display During Gameplay**



**Player Winning**



**Endgame Statistics with a message**

