School of Mechanical & Manufacturing Engineering (SMME),

National University of Science and Technology (NUST),

Sector H-12, Islamabad

Program: BE-Aerospace **Section:** AE-01

Session: Fall 2023 **Semester**: 1st

Course Title: Fundamentals of Programming (CS-109)

Tic Tac Toe Project

Name: Ayesha Ayaz

CMS: 481761

Playful Grids

Creating a tic tae toe program from scratch

Creating my Tic-Tac-Toe game in C++ involves designing a program that allows me, the user, to play against an AI opponent, taking turns marking spaces in a 3x3 grid. The ultimate goal is to achieve a winning pattern of three marks in a row, column, or diagonal.

Here's how I formulated and structured the code, ensuring it functioned successfully in gameplay.

```
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

char board[3][3] = {{' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '}};
bool isBoardFull() {
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (board[i][j] == ' ')
                return false;
        }}
    return true;
}
void displayBoard() {
    cout << "-------------" << endl;
    for (int i = 0; i < 3; ++i) {
        cout << "| ";
        for (int j = 0; j < 3; ++j) {
            cout << board[i][j] << " | ";
        }
        cout << endl;
```

```
        cout << "-------------" << endl;
    }}
bool checkWin(char symbol) {
    for (int i = 0; i < 3; ++i) {
        if ((board[i][0] == symbol && board[i][1] == symbol && board[i][2] == symbol) ||
            (board[0][i] == symbol && board[1][i] == symbol && board[2][i] == symbol))
            return true;
    }
    if ((board[0][0] == symbol && board[1][1] == symbol && board[2][2] == symbol) ||
        (board[0][2] == symbol && board[1][1] == symbol && board[2][0] == symbol))
        return true;
    return false;
}
void playerMove(char symbol) {
    int row, col;
    while (true) {
        if (symbol == 'X') {
            cout << "Player '" << symbol << "', enter your move (row and column): ";
            cin >> row >> col;
        } else {
            row = rand() % 3;
            col = rand() % 3;
        }
        if (row < 0 || row > 2 || col < 0 || col > 2 || board[row][col] != ' ') {
            cout << "Invalid move. Try again." << endl;
        } else {
            board[row][col] = symbol;
            break;
        }    }}
```

```
int main() {
    srand(static_cast<unsigned int>(time(0)));
    cout << "Welcome to Tic-Tac-Toe against AI!\n";
    char currentPlayer = 'X';

    while (true) {
        displayBoard();
        playerMove(currentPlayer);

        if (checkWin(currentPlayer)) {
            displayBoard();
            if (currentPlayer == 'X') {
                cout << "Player 'X' wins! Congratulations!" << endl;
            } else {
                cout << "AI wins! Better luck next time!" << endl;
            }
            break;
        }
        if (isBoardFull()) {
            displayBoard();
            cout << "It's a draw! Game over." << endl;
            break;
        }
        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
    }
    return 0;
}
```

Explanation of Code:

Board Initialization: A 3x3 grid is created as the game board using a 2D array (char board [3][3]).

Functions Defined:

isBoardFull(): Checks if the board is fully occupied.

displayBoard(): Renders the current state of the board.

checkWin(char symbol): Verifies if a player has won by examining rows, columns, and diagonals.

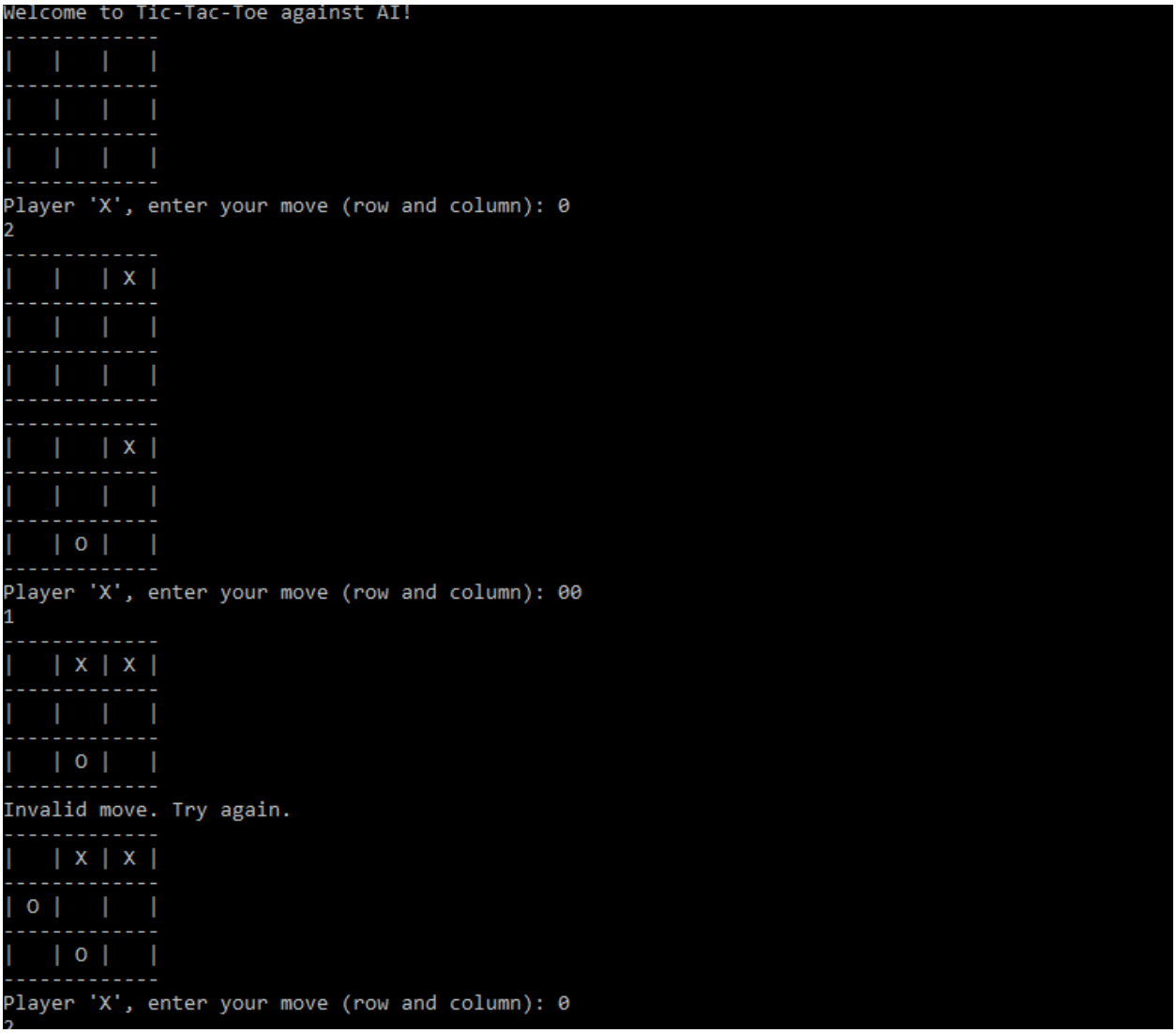
playerMove(char symbol): Manages player moves. For the AI, it generates random moves.

Main Function:

Initiates the game loop, alternating between the player and AI's moves.

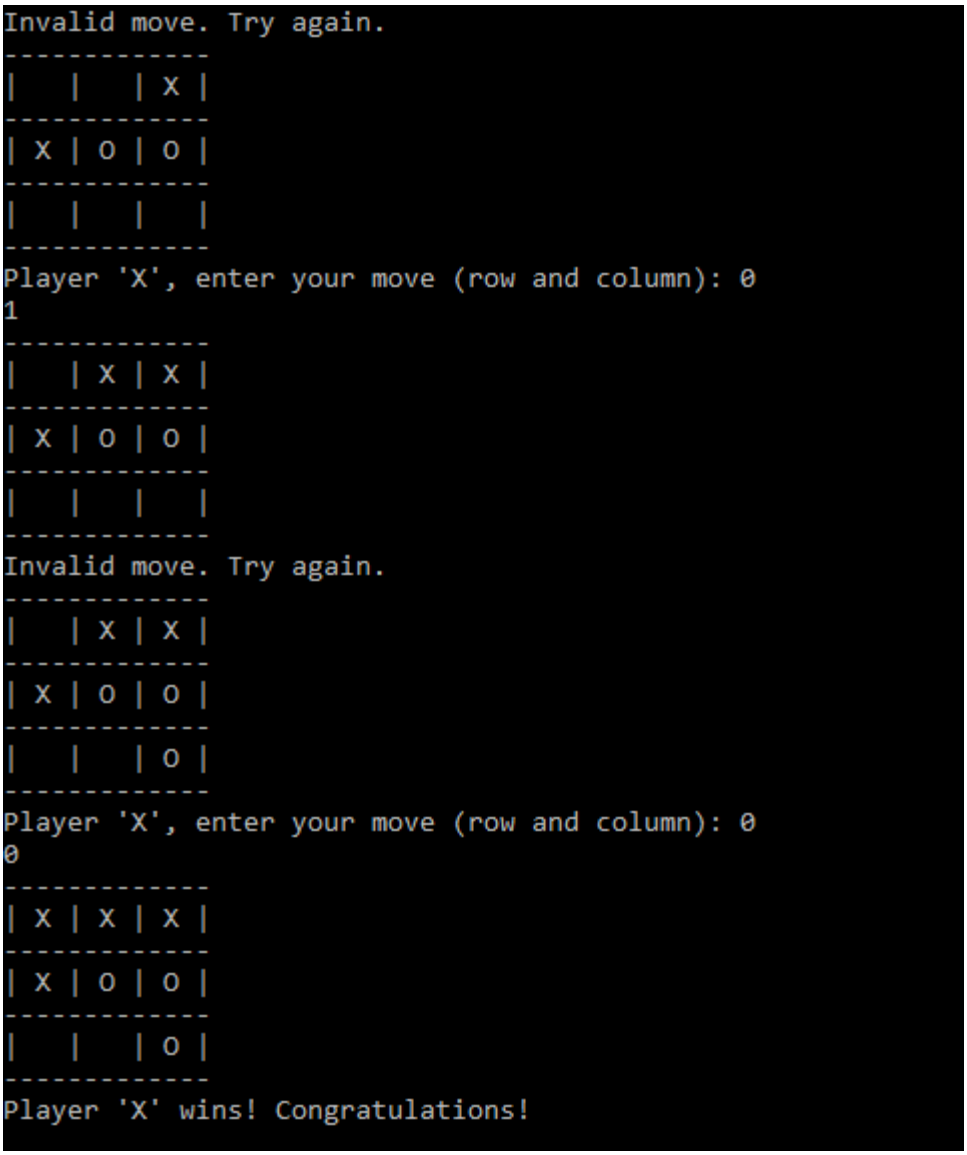
Ends the game when a player wins or when the board is full.

Screenshots of different outputs:



```
-------------
|   | X | X |
-------------
|   |   |   |
-------------
|   | O |   |
-------------
Invalid move. Try again.
-------------
|   | X | X |
-------------
| O |   |   |
-------------
|   | O |   |
-------------
Player 'X', enter your move (row and column): 0
2
Invalid move. Try again.
Player 'X', enter your move (row and column): 0
1
Invalid move. Try again.
Player 'X', enter your move (row and column): 00
0
-------------
| X | X | X |
-------------
| O |   |   |
-------------
|   | O |   |
-------------
Player 'X' wins! Congratulations!
```

```
Welcome to Tic-Tac-Toe against AI!
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
Player 'X', enter your move (row and column): 0
2
-------------
|   |   | X |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
-------------
|   |   | X |
-------------
|   |   | O |
-------------
|   |   |   |
-------------
Player 'X', enter your move (row and column): 1
0
-------------
|   |   | X |
-------------
| X |   | O |
-------------
|   |   |   |
-------------
Invalid move. Try again.
```



Explanation of Outputs:

Gameplay Illustration:

Describes each step of the game, highlighting moves made by the player and AI.

Victory Messages Interpretation:

Explains the outcome messages displayed upon game conclusion.

Conclusion:

Recapitulates the significance of Tic-Tac-Toe as an introductory game and its relevance in strategy and entertainment.

This report encapsulates the code implementation, its functionalities, for the Tic-Tac-Toe game against an AI opponent, offering insights into its gameplay.