Name : Manaswi Kulkarni

Roll no. 47

Class : SE comp2

MINI PROJECT -

#include <iostream>

#include <vector>

#include <cstdlib> // For rand() and srand()

#include <ctime>   // For time()

using namespace std;

class TicTacToe {

private:

    vector<vector<char>> board;

    char currentPlayer;

    char humanPlayer;

    char computerPlayer;

public:

    TicTacToe() : board(3, vector<char>(3, ' ')) {

        srand(time(0)); // Seed the random number generator

    }

    void printBoard() {

        cout << "   0 \t  1 \t  2\n";

        for (int i = 0; i < 3; i++) {

            cout << i << " ";

            for (int j = 0; j < 3; j++) {

                char mark = board[i][j];

                if (mark == 'X') {

                    cout << "\033[31m" << mark << "\033[0m"; // Red for X

                } else if (mark == 'O') {

                    cout << "\033[33m" << mark << "\033[0m"; // Yellow for O

                } else {

                    cout << mark;

                }

                if (j < 2) cout << "    | ";

            }

            cout << endl;

            if (i < 2) cout << "  -------------------\n";

        }

    }

    bool placeMark(int row, int col) {

        if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == ' ') {

            board[row][col] = currentPlayer;

            return true;

        }

        return false;

    }

    bool checkWin() {

        // Check rows and columns

        for (int i = 0; i < 3; i++) {

            if ((board[i][0] == currentPlayer && board[i][1] == currentPlayer && board[i][2] == currentPlayer) ||

                (board[0][i] == currentPlayer && board[1][i] == currentPlayer && board[2][i] == currentPlayer)) {

                return true;

            }

        }

        // Check diagonals

        if ((board[0][0] == currentPlayer && board[1][1] == currentPlayer && board[2][2] == currentPlayer) ||

            (board[0][2] == currentPlayer && board[1][1] == currentPlayer && board[2][0] == currentPlayer)) {

            return true;

        }

        return false;

    }

    bool checkDraw() {

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                if (board[i][j] == ' ') {

                    return false;

                }

            }

        }

        return true;

    }

    void switchPlayer() {

        currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';

    }

    // Function to get a random move for the computer

    void getRandomMove(int &row, int &col) {

        do {

            row = rand() % 3;

            col = rand() % 3;

        } while (board[row][col] != ' ');

    }

    void playGame() {

        // Ask the user to choose between X and O

        cout << "Do you want to be X or O? ";

        cin >> humanPlayer;

        // Validate the input

        while (humanPlayer != 'X' && humanPlayer != 'O') {

            cout << "Invalid choice. Please choose X or O: ";

            cin >> humanPlayer;

        }

        computerPlayer = (humanPlayer == 'X') ? 'O' : 'X';

        currentPlayer = 'X'; // X always starts the game

        int row, col;

        while (true) {

            printBoard();

            if (currentPlayer == humanPlayer) {

                // Human player's turn

                cout << "Player " << currentPlayer << ", enter your move (row and column): ";

                cin >> row >> col;

            } else {

                // Computer player's turn

                cout << "Computer is making a move..." << endl;

                getRandomMove(row, col);

            }

            if (placeMark(row, col)) {

                if (checkWin()) {

                    printBoard();

                    cout << "Player " << currentPlayer << " wins!" << endl;

                    break;

                }

                if (checkDraw()) {

                    printBoard();

                    cout << "It's a draw!" << endl;

                    break;

                }

                switchPlayer();

            } else {

                if (currentPlayer == humanPlayer) {

                    cout << "Invalid move. Try again." << endl;

                }

            }

        }

    }

};

int main() {

    TicTacToe game;

    game.playGame();

    return 0;

}

OUTPUT

OUTPUT

Do you want to be X or O? X

0 1 2

0 | |

-------------------

1 | |

-------------------

2 | |

Player X, enter your move (row and column): 1

1

0 1 2

0 | |

-------------------

1 | X |

-------------------

2 | |

Computer is making a move...

0 1 2

0 | |

-------------------

1 O | X |

-------------------

2 | |

Player X, enter your move (row and column): 2

1

0 1 2

0 | |

-------------------

1 O | X |

-------------------

2 | X |

Computer is making a move...

0 1 2

0 | |

-------------------

1 O | X | O

-------------------

2 | X |

Player X, enter your move (row and column): 1

0

Invalid move. Try again.

0 1 2

0 | |

-------------------

1 O | X | O

-------------------

2 | X |

Player X, enter your move (row and column): 2

2

0 1 2

0 | |

-------------------

1 O | X | O

-------------------

2 | X | X

Computer is making a move...

0 1 2

0 | |

-------------------

1 O | X | O

-------------------

2 O | X | X

Player X, enter your move (row and column): 0

0

0 1 2

0 X | |

-------------------

1 O | X | O

-------------------

2 O | X | X

Player X wins!