Assignment Topic: Tic-tac-toe Implementation with Min-max Algorithm

Course Title: Artificial Intelligence

Course Code: SWE 323

Date of Submission: 31st May 2023

Submitted to:

Sayma Sultana Chowdhury

Assistant Professor, IICT, SUST

Submitted by:

Sumonta Saha Mridul

Reg No: 2019831056

IICT, SUST

```
#include <iostream>
#include <vector>

using namespace std;

struct Move {
    int row, col;
};

char player = 'x', opponent = 'o';

bool isMovesLeft(const vector<vector<char>>& board) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] == '_') {
                return true;
            }
        }
    }
    return false;
}

int evaluate(const vector<vector<char>>& board) {
    for (int row = 0; row < 3; row++) {
        if (board[row][0] == board[row][1] && board[row][1] == board[row][2]) {
            if (board[row][0] == player)
                return +10;
            else if (board[row][0] == opponent)
                return -10;
        }
    }

    for (int col = 0; col < 3; col++) {
        if (board[0][col] == board[1][col] && board[1][col] == board[2][col]) {
            if (board[0][col] == player)
                return +10;
            else if (board[0][col] == opponent)
                return -10;
        }
    }

    if (board[0][0] == board[1][1] && board[1][1] == board[2][2]) {
        if (board[0][0] == player)
            return +10;
        else if (board[0][0] == opponent)
            return -10;
    }

    if (board[0][2] == board[1][1] && board[1][1] == board[2][0]) {
        if (board[0][2] == player)
            return +10;
        else if (board[0][2] == opponent)
            return -10;
    }

    return 0;
}

int minimax(vector<vector<char>>& board, int depth, bool isMax) {
    int score = evaluate(board);
    if (score == 10)
        return score;

    if (score == -10)
        return score;

    if (!isMovesLeft(board))
        return 0;

    if (isMax) {
        int best = -1000;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j] == '_') {
                    board[i][j] = player;
                    best = max(best, minimax(board, depth + 1, !isMax));
                    board[i][j] = '_';
                }
            }
        }
        return best;
    } else {
        int best = 1000;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
                if (board[i][j] == '_') {
                    board[i][j] = opponent;
                    best = min(best, minimax(board, depth + 1, !isMax));
                    board[i][j] = '_';
                }
            }
        }
        return best;
    }
}

Move findBestMove(vector<vector<char>>& board) {
    int bestVal = -1000;
    Move bestMove;
    bestMove.row = -1;
    bestMove.col = -1;

    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            if (board[i][j] == '_') {
                board[i][j] = player;
                int moveVal = minimax(board, 0, false);
                board[i][j] = '_';
                if (moveVal > bestVal) {
                    bestMove.row = i;
                    bestMove.col = j;
                    bestVal = moveVal;
                }
            }
        }
    }

    cout << "The value of the best move is: " << bestVal << endl << endl;
    return bestMove;
}
```

```
void printBoard(const vector<vector<char>>& board) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << board[i][j] << " ";
        }
        cout << endl;
    }
}

int main() {
    vector<vector<char>> board(3, vector<char>(3, '_'));

    cout << "Enter the initial state of the board:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cin >> board[i][j];
        }
    }

    Move bestMove = findBestMove(board);

    cout << "The optimal move is:" << endl;
    cout << "ROW: " << bestMove.row << " COL: " << bestMove.col << endl << endl;

    cout << "Updated board state:" << endl;
    board[bestMove.row][bestMove.col] = player;
    printBoard(board);

    return 0;
}
```