Assignment 1

AI Tic Tak Toe :

Code :

#include <iostream>

using namespace std;

const char PLAYER\_X = 'X';

const char PLAYER\_O = 'O';

const char EMPTY = ' ';

bool isdraw(char a[][3]);

int ttt(char a[][3]);

int minimax(char a[][3], int depth, bool isMaximizing);

void printBoard(const char a[][3]);

bool isGameOver(const char a[][3]) {

// Check rows, columns, and diagonals for a winner

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

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

return true;

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

return true;

}

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

return true;

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

return true;

// Check if the board is full (draw)

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

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

if (a[i][j] == ' ')

return false;

}

}

// If there is no winner and the board is full, it's a draw

return true;

}

bool iswin(const char a[][3]) {

// Check rows, columns, and diagonals for a winner

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

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

return true;

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

return true;

}

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

return true;

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

return true;

// If there is no winner and the board is full, it's a draw

return false;

}

int evaluateBoard(const char a[][3]) {

// Check rows, columns, and diagonals for a possible win

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

if (a[i][0] == a[i][1] && a[i][1] == a[i][2]) {

if (a[i][0] == PLAYER\_X)

return 10;

else if (a[i][0] == PLAYER\_O)

return -10;

}

if (a[0][i] == a[1][i] && a[1][i] == a[2][i]) {

if (a[0][i] == PLAYER\_X)

return 10;

else if (a[0][i] == PLAYER\_O)

return -10;

}

}

if (a[0][0] == a[1][1] && a[1][1] == a[2][2]) {

if (a[0][0] == PLAYER\_X)

return 10;

else if (a[0][0] == PLAYER\_O)

return -10;

}

if (a[0][2] == a[1][1] && a[1][1] == a[2][0]) {

if (a[0][2] == PLAYER\_X)

return 10;

else if (a[0][2] == PLAYER\_O)

return -10;

}

return 0; // No winner yet

}

int minimax(char a[][3], int depth, bool isMaximizing) {

int bestScore=0;

if (isGameOver(a)){

//cout<<evaluateBoard(a)<<endl;

return evaluateBoard(a);}

if (isMaximizing) {

bestScore = -1000;

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

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

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

a[i][j] = PLAYER\_X;

//cout<<i<<" "<<j<<endl;

int score = minimax(a, depth + 1, false);

a[i][j] = ' ';

bestScore = max(bestScore, score);

}

}

}

return bestScore;

} else {

bestScore = 1000;

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

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

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

a[i][j] = PLAYER\_O;

int score = minimax(a, depth + 1, true);

a[i][j] = ' ';

bestScore = min(bestScore, score);

}

}

}

return bestScore;

}

}

void findBestMove(char a[][3]) {

int bestScore = -1000;

int bestMoveRow = -1;

int bestMoveCol = -1;

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

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

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

a[i][j] = PLAYER\_X;

int score = minimax(a, 0, false);

a[i][j] = ' ';

if (score > bestScore) {

bestScore = score;

//cout<<bestScore<<endl;

bestMoveRow = i;

bestMoveCol = j;

}

}

}

}

a[bestMoveRow][bestMoveCol] = PLAYER\_X;

}

void printBoard(const char a[][3]) {

cout << " -------------" << endl;

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

cout << " | ";

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

cout << a[i][j] << " | ";

}

cout << endl;

cout << " -------------" << endl;

}

}

int ttt(char a[][3]) {

int t = 1;

int p;

int result = 0;

while (!isdraw(a)) {

if (t == 1) {

cout << "Player " << t << "'s turn: " << endl;

cout << "Enter the position: " << endl;

cin >> p;

int row = (p - 1) / 3;

int col = (p - 1) % 3;

if (a[row][col] == ' ') {

a[row][col] = PLAYER\_O;

printBoard(a);

if(!isGameOver(a)){

if (iswin(a)) {

return 1; // Player 1 wins

} else {

t = 2; // Player 2 (AI) turn

}

}

} else {

cout << "Position " << p << " is already filled" << endl;

}

} else {

findBestMove(a);

cout << "AI's move:" << endl;

printBoard(a);

//cout<<"@";

if (iswin(a)) {

return 2; // AI wins

} else {

t = 1; // Player 1 turn

}

}

}

return 0; // 0 for draw

}

bool isdraw(char a[][3]) {

// Check if the board is full (draw)

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

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

if (a[i][j] == ' ')

return false;

}

}

// Check if there is no winner

return true;

}

int main() {

char a[3][3] = { {' ', ' ', ' '}, {' ', ' ', ' '}, {' ', ' ', ' '} };

//char a[3][3] = { {'0','0','X'}, {'X','X','0'}, {'0','X','0'} };

cout << " Welcome to Tic Tac Toe" << endl;

cout << " Rules:" << endl;

cout << " 1) Player 1 has 'O' and AI has 'X'" << endl;

cout << " 2) First Enter the position number and then press enter" << endl;

cout << " 3) Position numbers are as follows:" << endl;

cout << endl;

int k = 1;

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

cout << " -------------" << endl;

cout << " | ";

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

cout << k << " | ";

k++;

}

cout << endl;

}

cout << " -------------" << endl;

cout << endl;

cout<<"Board"<<endl;

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

cout << " -------------" << endl;

cout << " | ";

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

cout <<" " << " | ";

k++;

}

cout << endl;

}

cout << " -------------" << endl;

cout << endl;

int result = ttt(a);

if (result == 1) {

cout << "Player 1 wins!" << endl;

} else if (result == 2) {

cout << "(AI) wins!" << endl;

} else {

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

}

return 0;

}