***Tic tac toe GAME***

***MINI PROJECT***

***BY-AMEYAGARG***

***RA2111033010090***

***DESCRIPTION-***

Tic-tac-toe is a game where**two players X and O fill the hash (#) shaped box** (consist of two vertical lines crossing two horizontal lines) with their alternate turns. The player who first fills the box with 3Xs or 3Os in a horizontal, vertical, or diagonal manner will win the game.

This game is made for single and dual player. Where in single player the code will play with you.

***ALGORITHM-***

* **STEP1-the code will ask the number of players.**
* **Step2-then according to the number of players it will ask the player/players name.**
* **Step3-then the code will ask to select the (x or o) for each player.**
* **Step4- then a new (#) shaped will be formed with a sequenced number from 1 to 9.**
* **Step5-then simultaneously the player has to fill the number according to the move to defeat the opponent**
* **Step6- if the symmetry of 3xs or 3os are formed so the first player would be declared as winner else no symmetry is formed so it will show ‘draw’.**
* **End.**

***CODE***

#include <ctype.h>

#include <math.h>

#include <stdbool.h>

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <time.h>

typedef struct

{

char name[30];

char symbol;

}

player;

const int N = 3;

char board[3][3] = {

{'1', '2', '3'},

{'4', '5', '6'},

{'7', '8', '9'}

};

player players[2];

char COMPUTER\_AI;

int numOfplayer, winner;

void delay(int seconds);

void information();

void draw\_board();

void draw\_line();

void two\_player\_gameplay();

int check\_win();

int check\_minimax\_win();

void show\_the\_winner(int result);

void human\_move();

void computer\_move();

int minimax(int depth, bool isMaximizing);

int main(void)

{

printf("Welcome to Tic-Tac-Toe game!\n\n");

information();

if (numOfplayer == 2)

{

two\_player\_gameplay();

}

else

{

if (players[0].symbol == 'X')

{

human\_move();

}

else

{

computer\_move();

}

}

return 0;

}

void human\_move()

{

int choice, found, result;

char mark, char\_choice;

draw\_board();

do

{

printf("\t\t\t\t");

printf("%s : ", players[0].name);

scanf("%i", &choice);

mark = players[0].symbol;

found = 0;

for (int row = 0; row < N; row++)

{

for (int col = 0; col < N; col++)

{

char\_choice = choice + '0';

if (board[row][col] == char\_choice)

{

board[row][col] = mark;

found++;

row = col = N;

break;

}

}

}

}

while ((choice < 1 || choice > 9) || found == 0);

result = check\_win();

if (result == 2)

{

computer\_move();

}

else

{

show\_the\_winner(result);

}

}

void computer\_move()

{

char board\_number;

int move\_i, move\_j;

float bestScore = -INFINITY;

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

{

if (board[i][j] != 'X' && board[i][j] != 'O')

{

board\_number = board[i][j];

board[i][j] = COMPUTER\_AI;

float score = minimax(0, false);

board[i][j] = board\_number;

if (score > bestScore)

{

bestScore = score;

move\_i = i;

move\_j = j;

}

}

}

}

board[move\_i][move\_j] = COMPUTER\_AI;

int result = check\_win();

if (result == 2)

{

human\_move();

}

else

{

show\_the\_winner(result);

}

}

int minimax(int depth, bool isMaximizing)

{

int result = check\_minimax\_win();

if (result != 2)

{

return result;

}

if (isMaximizing)

{

float bestScore = -INFINITY;

for (int row = 0; row < N; row++)

{

for (int col = 0; col < N; col++)

{

if (board[row][col] != 'X' && board[row][col] != 'O')

{

char board\_number = board[row][col];

board[row][col] = COMPUTER\_AI;

float score = minimax(depth + 1, false);

board[row][col] = board\_number;

if (score > bestScore)

{

bestScore = score;

}

}

}

}

return bestScore;

}

else

{

float bestScore = INFINITY;

for (int row = 0; row < N; row++)

{

for (int col = 0; col < N; col++)

{

if (board[row][col] != 'X' && board[row][col] != 'O')

{

char board\_number = board[row][col];

board[row][col] = players[0].symbol;

float score = minimax(depth + 1, true);

board[row][col] = board\_number;

if (score < bestScore)

{

bestScore = score;

}

}

}

}

return bestScore;

}

}

void two\_player\_gameplay()

{

int player = 0;

int choice, found, result;

char mark, char\_choice;

do

{

draw\_board();

printf("\t\t\t\t");

printf("%s : ", players[player % 2].name);

scanf("%i", &choice);

mark = players[player % 2].symbol;

for (int row = 0; row < N; row++)

{

for (int col = 0; col < N; col++)

{

char\_choice = choice + '0';

if (board[row][col] == char\_choice)

{

board[row][col] = mark;

player++;

row = col = 3;

break;

}

}

}

result = check\_win();

}

while ((choice < 1 || choice > 9) || result == 2);

draw\_board();

show\_the\_winner(result);

}

void information()

{

do

{

printf("How many player will play (MAX is 2): ");

scanf("%i", &numOfplayer);

}

while (numOfplayer > 2);

printf("\n");

if(numOfplayer < 2)

{

printf("Enter your name: ");

scanf("%s", players[0].name);

getchar();

do

{

printf("X or O: ");

scanf("%c", &players[0].symbol);

getchar();

}

while (players[0].symbol != 'X' && players[0].symbol != 'O');

players[0].symbol = toupper(players[0].symbol);

COMPUTER\_AI = (players[0].symbol == 'X' ? 'O' : 'X');

}

else

{

for (int i = 0; i < 2; i++)

{

printf("Enter Player %i name: ", i + 1);

scanf("%s", players[i].name);

getchar();

printf("\n");

}

players[0].symbol = toupper('X');

players[1].symbol = ((toupper(players[0].symbol) == 'X') ? 'O' : 'X');

}

printf("\n\n");

}

int check\_win()

{

for (int i = 0; i < N; i++)

{

for (int j = 1; j <= 1; j++)

{

if (board[i][j - 1] == board[i][j] && board[i][j] == board[i][j + 1])

{

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

{

return 1;

}

else

{

return -1;

}

}

else if (board[j - 1][i] == board[j][i] && board[j][i] == board[j + 1][i])

{

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

{

return 1;

}

else

{

return -1;

}

}

else if (board[j - 1][j - 1] == board[j][j] && board[j][j] == board[j + 1][j + 1])

{

if (board[j][j] == 'X')

{

return 1;

}

else

{

return -1;

}

}

else if (board[j - 1][j + 1] == board[j][j] && board[j][j] == board[j + 1][j - 1])

{

if (board[j][j] == 'X')

{

return 1;

}

else

{

return -1;

}

}

else if (board[j - 1][j - 1] != '1' && board[j - 1][j] != '2' && board[j - 1][j + 1] != '3' &&

board[j][j - 1] != '4' && board[j][j] != '5' && board[j][j + 1] != '6' &&

board[j + 1][j - 1] != '7' && board[j + 1][j] != '8' && board[j + 1][j + 1] != '9')

{

return 0;

}

}

}

return 2;

}

int check\_minimax\_win()

{

for (int i = 0; i < N; i++)

{

for (int j = 1; j <= 1; j++)

{

if (board[i][j - 1] == board[i][j] && board[i][j] == board[i][j + 1])

{

if (players[0].symbol == 'O')

{

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

{

return 1;

}

else

{

return -1;

}

}

if (players[0].symbol == 'X')

{

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

{

return -1;

}

else

{

return 1;

}

}

}

else if (board[j - 1][i] == board[j][i] && board[j][i] == board[j + 1][i])

{

if (players[0].symbol == 'O')

{

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

{

return 1;

}

else

{

return -1;

}

}

if (players[0].symbol == 'X')

{

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

{

return -1;

}

else

{

return 1;

}

}

}

else if (board[j - 1][j - 1] == board[j][j] && board[j][j] == board[j + 1][j + 1])

{

if (players[0].symbol == 'O')

{

if (board[j][j] == 'X')

{

return 1;

}

else

{

return -1;

}

}

if (players[0].symbol == 'X')

{

if (board[j][j] == 'X')

{

return -1;

}

else

{

return 1;

}

}

}

else if (board[j - 1][j + 1] == board[j][j] && board[j][j] == board[j + 1][j - 1])

{

if (players[0].symbol == 'O')

{

if (board[j][j] == 'X')

{

return 1;

}

else

{

return -1;

}

}

if (players[0].symbol == 'X')

{

if (board[j][j] == 'X')

{

return -1;

}

else

{

return 1;

}

}

}

else if (board[j - 1][j - 1] != '1' && board[j - 1][j] != '2' && board[j - 1][j + 1] != '3' &&

board[j][j - 1] != '4' && board[j][j] != '5' && board[j][j + 1] != '6' &&

board[j + 1][j - 1] != '7' && board[j + 1][j] != '8' && board[j + 1][j + 1] != '9')

{

return 0;

}

}

}

return 2;

}

void show\_the\_winner(int result)

{

if (numOfplayer < 2)

{

if ((players[0].symbol == 'X' && result == 1) || (players[0].symbol == 'O' && result == -1))

{

draw\_board();

printf("\n\t\t\t\t\t ---THE WINNER IS %s---\n\n\n", players[0].name);

}

else if (result == -1 || result == 1)

{

draw\_board();

printf("\n\t\t\t\t\t\t ---YOU LOSE---\n\n\n");

}

else

{

draw\_board();

printf("\n\t\t\t\t\t ---THE GAME IS DRAW---\n\n\n");

}

}

else

{

for (int i = 0; i < 2; i++)

{

if (result == 1)

{

if (players[i].symbol == 'X')

{

printf("\n\t\t\t\t\t ---THE WINNER IS %s---\n\n\n", players[i].name);

break;

}

}

if (result == -1)

{

if (players[i].symbol == 'O')

{

printf("\n\t\t\t\t\t ---THE WINNER IS %s---\n\n\n", players[i].name);

break;

}

}

if (result == 0)

{

printf("\n\t\t\t\t\t ---THE GAME IS DRAW---\n\n\n");

break;

}

}

}

}

void draw\_board()

{

system("clear");

printf("\n\n");

printf("\t\t\t\tEnter the number on board to choose your place!\n\n\n");

if (numOfplayer < 2)

{

printf("\t\t\t\t");

printf("%s : %c", players[0].name, toupper(players[0].symbol));

}

else

{

printf("\t\t\t\t");

printf("(Player 1) %s : %c\n", players[0].name, toupper(players[0].symbol));

printf("\t\t\t\t");

printf("(Player 2) %s : %c\n", players[1].name, toupper(players[1].symbol));

}

printf("\n\n");

for (int row = 0; row < N; row++)

{

for (int col = 0; col < N; col++)

{

if (col == 0)

{

printf("\t\t\t\t\t\t");

printf(" | | ");

printf("\n");

printf("\t\t\t\t\t\t");

}

printf(" %c ", board[row][col]);

if (col != 2)

{

printf("|");

}

}

printf("\n");

if (row != 2)

{

printf("\t\t\t\t\t\t");

printf("\_\_\_\_\_|\_\_\_\_\_|\_\_\_\_\_");

printf("\n");

}

if (row == 2)

{

printf("\t\t\t\t\t\t");

printf(" | | ");

printf("\n");

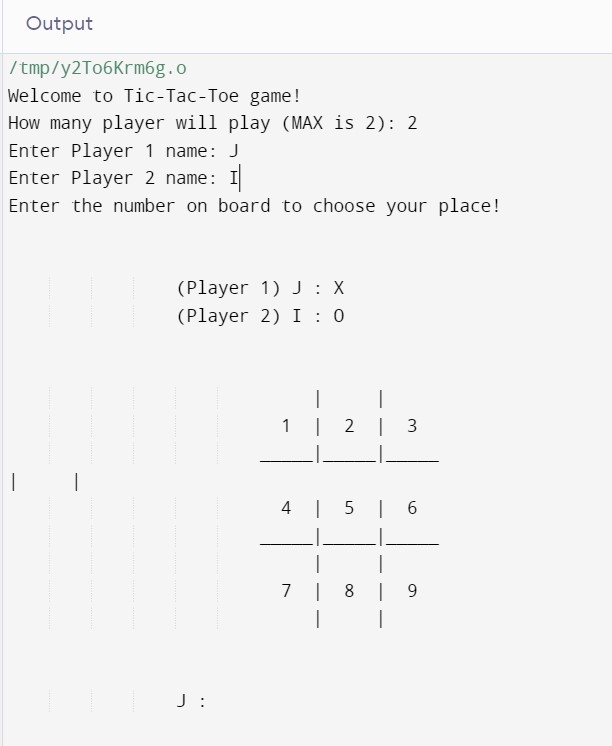
}

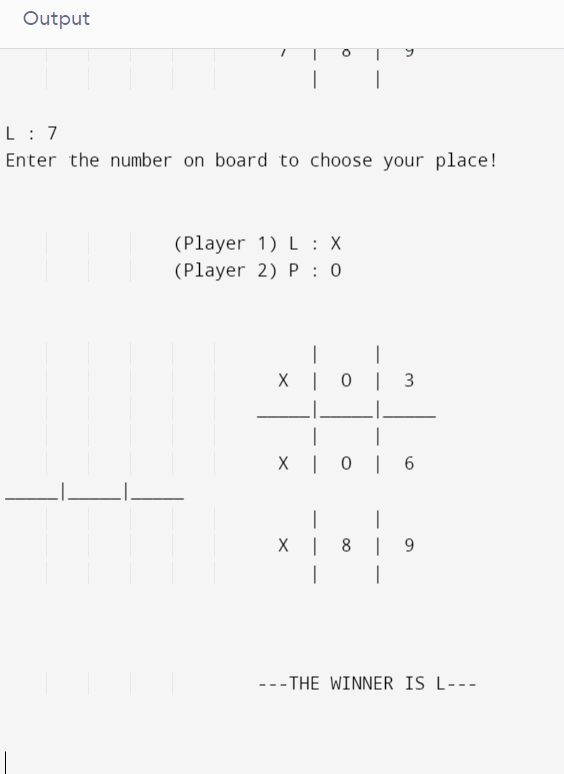
}

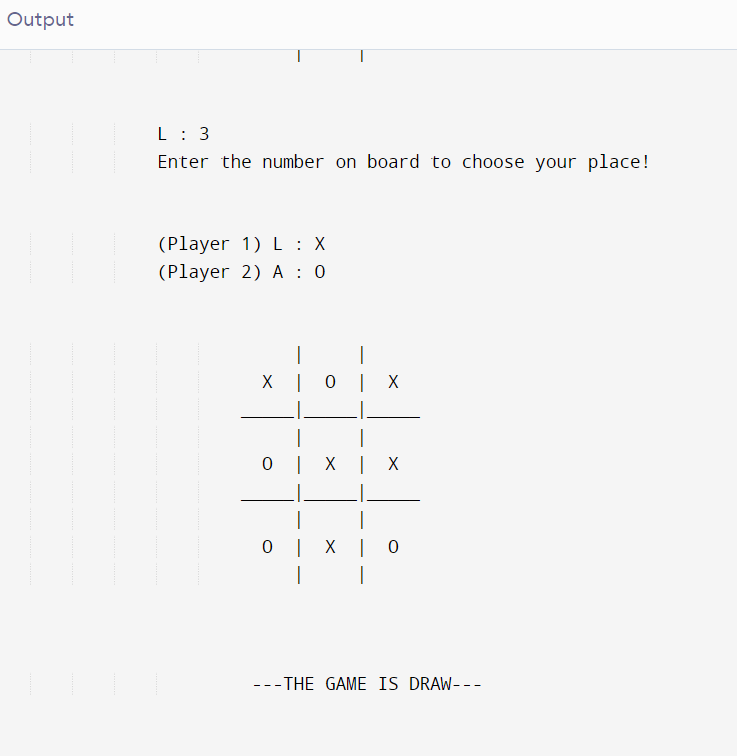
printf("\n\n");

}

***OUTPUT***

**

**

**