Roll No. 50

Assignment 1 : Minimax and Posswin

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

**1) Posswin:-**

#include<stdio.h>

#include<stdlib.h>

#include<string.h>

#include<math.h>

#define S 9

#define b 2

#define X 3

#define O 5

#define row 1

#define col 2

#define diag 3

#define Xwin 100

#define Owin 200

int dimension;

int Grid[S];

int valid()

{

int ver;

char c;

while(1)

{

if(scanf("%d%c", &ver, &c) == 2 && c == '\n')

{

return ver;

}

else

{

printf("Invalid Input, Try Again.\n");

fflush(stdin);

}

}

}

void input(int P)

{

int x;

printf("User's Turn:\nEnter number to fill:\n");

while(!((x = valid()) > 0 && x <= S && Grid[x-1] == b))

{

printf("Invalid Input, Try Again.\n");

}

Grid[x - 1] = P;

}

void initialize()

{

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

Grid[i] = b;

dimension = sqrt(S);

}

void disp()

{

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

{

if(!(i%dimension))

printf("\n\n");

if(Grid[i] == X)

printf("X\t");

else if(Grid[i] == O)

printf("O\t");

else

printf("%d\t", i+1);

}

printf("\n\n");

}

int mult(int count, int type)

{

if(type == row)

{

return Grid[count]\*Grid[count+1]\*Grid[count+2];

}

else if(type == col)

{

return Grid[count]\*Grid[count+dimension]\*Grid[count+2\*dimension];

}

else

return -1;

}

int position(int count, int type)

{

if(type == row)

{

for(int i = count; i < count + dimension; i++)

{

if(Grid[i] == b)

return i;

}

}

else if(type == col)

{

for(int i = 0, j = count; i < dimension; i++, j = j + dimension)

{

if(Grid[j] == b)

return j;

}

}

else if(type == diag)

{

if(count == 1)

{

for(int i = 0; i < 9; i = i + 4)

{

if(Grid[i] == b)

return i;

}

}

if(count == 2)

{

for(int i = 2; i <= 6; i = i + 2)

{

if(Grid[i] == b)

return i;

}

}

}

return -1;

}

int posswin(int P)

{

int uno = X\*X\*b, dos = O\*O\*b, win1 = X\*X\*X, win2 = O\*O\*O;

int prodx = 1, prody = 1, prod1, prod2;

prod1 = Grid[0]\*Grid[4]\*Grid[8];

prod2 = Grid[2]\*Grid[4]\*Grid[6];

if(P == X)

{

if(prod1 == uno)

{

return position(1, diag);

}

if(prod2 == uno)

{

return position(2, diag);

}

if(prod1 == win1)

{

return Xwin;

}

if(prod2 == win1)

{

return Xwin;

}

else

{

for(int i = 0, j = 0; i < dimension; i++, j = j + dimension)

{

if(mult(j, row) == uno)

{

return position(j, row);

}

if(mult(j, row) == win1)

{

return Xwin;

}

}

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

{

if(mult(i, col) == uno)

{

return position(i, col);

}

if(mult(i, col) == win1)

{

return Xwin;

}

}

}

}

else if(P == O)

{

if(prod1 == dos)

{

return position(1, diag);

}

if(prod2 == dos)

{

return position(2, diag);

}

if(prod1 == win2)

{

return Owin;

}

if(prod2 == win2)

{

return Owin;

}

for(int i = 0, j = 0; i < dimension; i++, j = j + dimension)

{

if(mult(j, row) == dos)

{

return position(j, row);

}

if(mult(j, row) == win2)

{

return Owin;

}

}

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

{

if(mult(i, col) == dos)

{

return position(i, col);

}

if(mult(i, col) == win2)

{

return Owin;

}

}

}

return -1;

}

void cpu(int a)

{

printf("CPU Marked at %d.\n", a + 1);

}

void winMsg(int P)

{

if(P == X)

printf("\nX has won the game !\n");

else

printf("\nO has won the game !\n");

}

void play(int P)

{

int x;

if(P == 1) // Player is X

{

// Turn 1 X

disp();

input(X);

system("cls");

// Turn 2 O

if(Grid[4] == b)

{

Grid[4] = O;

x = 4;

}

else

{

x = 0;

Grid[0] = O;

}

cpu(x);

// Turn 3 X

disp();

input(X);

system("cls");

// Turn 4 O

x = posswin(X);

if(x != -1)

{

Grid[x] = O;

}

else

{

if(Grid[0] == b)

{

x = 0;

Grid[0] = O;

}

else

{

x = 2;

Grid[2] = O;

}

}

cpu(x);

// Turn 5 X

disp();

input(X);

system("cls");

// Turn 6 O

x = posswin(O);

if(x != -1)

{

Grid[x] = O;

cpu(x);

disp();

winMsg(O);

return;

}

else

{

x = posswin(X);

if(x != -1)

{

Grid[x] = O;

}

else

{

if(Grid[0] == b)

{

x = 0;

Grid[0] = O;

}

else if(Grid[2] == b)

{

x = 2;

Grid[2] = O;

}

else

{

x = 8;

Grid[8] = O;

}

cpu(x);

}

}

// Turn 7 X

disp();

input(X);

if(posswin(X) == Xwin)

{

disp();

winMsg(X);

return;

}

system("cls");

// Turn 8 O

x = posswin(O);

if(x != -1)

{

Grid[x] = O;

cpu(x);

disp();

winMsg(O);

return;

}

else

{

x = posswin(X);

if(x != -1)

{

Grid[x] = O;

}

else

{

if(Grid[0] == b)

{

x = 0;

Grid[0] = O;

}

else if(Grid[2] == b)

{

x = 2;

Grid[2] = O;

}

else if(Grid[6] == b)

{

x = 6;

Grid[6] = O;

}

else if(Grid[1] == b)

{

x = 1;

Grid[1] = O;

}

else if(Grid[3] == b)

{

x = 3;

Grid[3] = O;

}

else

{

x = 7;

Grid[7] = O;

}

}

cpu(x);

}

// Turn 9 X

disp();

input(X);

disp();

printf("Draw\n");

} //end if

else // Player is O

{

// Turn 1 X

Grid[4] = X;

cpu(4);

// Turn 2 O

disp();

input(O);

system("cls");

// Turn 3 X

if(Grid[0] == b)

{

x = 0;

Grid[0] = X;

}

else

{

x = 2;

Grid[2] = X;

}

cpu(x);

// Turn 4 O

disp();

input(O);

system("cls");

// Turn 5 X

x = posswin(X);

if(x != -1)

{

Grid[x] = X;

cpu(x);

disp();

winMsg(X);

return;

}

else

{

x = posswin(O);

if(x != -1)

{

Grid[x] = X;

}

else

{

if(Grid[2] == b)

{

x = 2;

Grid[2] = X;

}

else

{

x = 8;

Grid[8] = X;

}

}

cpu(x);

}

// Turn 6 O

disp();

input(O);

/\*

disp();

printf("Press any key to continue");

getch();

\*/ system("cls");

// Turn 7 X

x = posswin(X);

if(x != -1)

{

Grid[x] = X;

cpu(x);

disp();

winMsg(X);

return;

}

else

{

if(Grid[2] == b)

{

x = 2;

Grid[2] = X;

}

else

{

x = 8;

Grid[8] = X;

}

cpu(x);

}

// Turn 8 O

disp();

input(O);

/\*

disp();

printf("Press any key to continue");

getch();

\*/ system("cls");

// Turn 9 X

x = posswin(X);

if(x != -1)

{

Grid[x] = X;

cpu(x);

disp();

winMsg(X);

return;

}

else

{

if(Grid[8] == b)

{

x = 8;

Grid[8] = X;

}

else if(Grid[1] == b)

{

x = 1;

Grid[1] = X;

}

else if(Grid[3] == b)

{

x = 3;

Grid[3] = X;

}

else if(Grid[5] == b)

{

x = 5;

Grid[5] = X;

}

else

{

x = 7;

Grid[7] = X;

}

cpu(x);

disp();

printf("\nDraw\n");

}

}

}

int main()

{

printf("\n------------------------------------------------------\n");

printf("\t\tTic");

printf(" Tac");

printf(" Toe");

printf("\n------------------------------------------------------\n");

initialize();

printf("\nChoose(1 or 2):\n1. X\n2. O\n");

int choice;

//disp();

while(!((choice = valid()) == 1 || choice == 2))

{

printf("Choose a correct option\n");

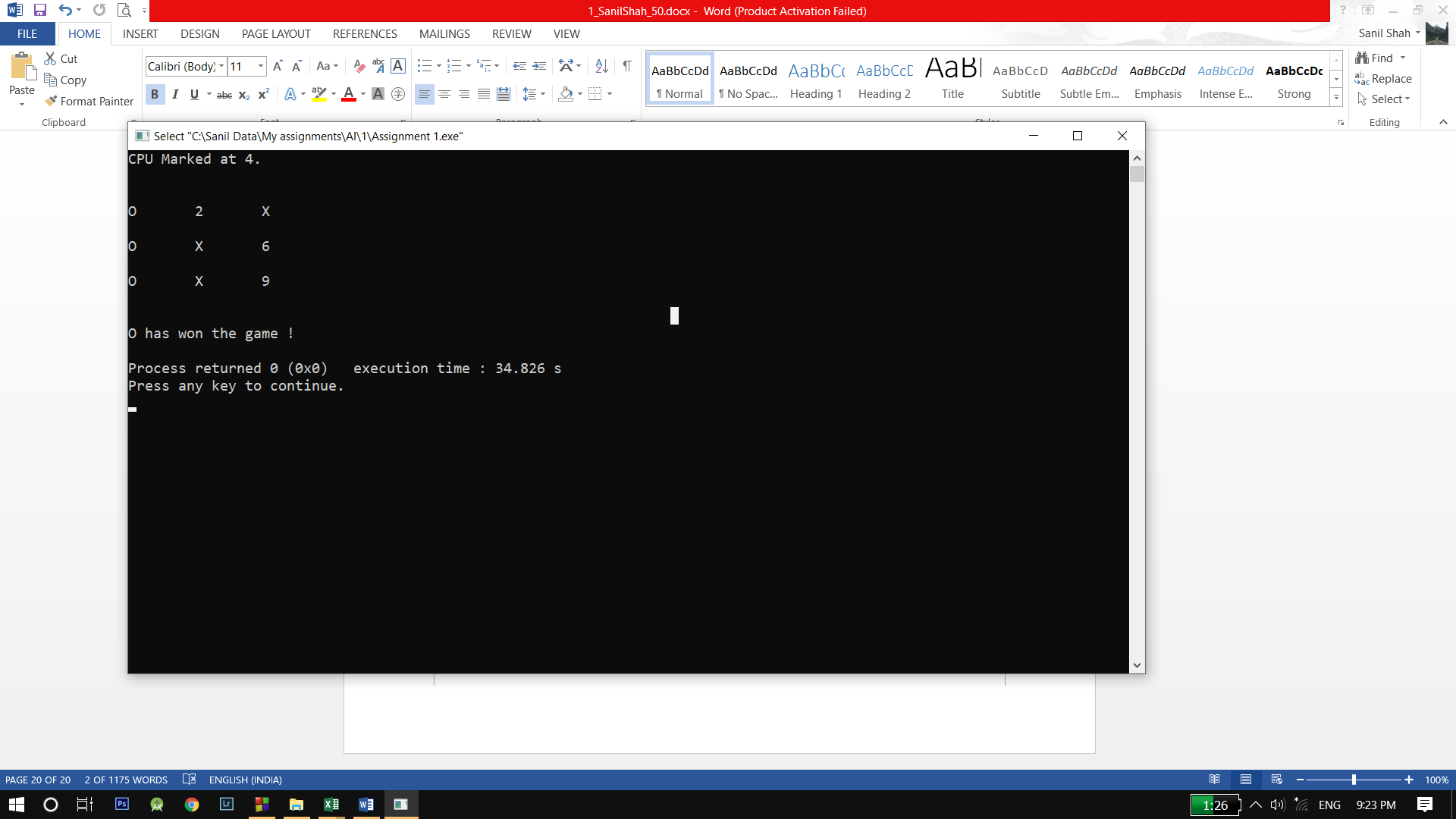
}

system("cls");

play(choice);

return 0;

}



**2) Minimax:**

// Minimax Algorithm in TicTacToe

#include <stdio.h>

char gridChar(int i) {

switch(i) {

case -1:

return 'X';

case 0:

return ' ';

case 1:

return 'O';

}

}

void draw(int b[9]) {

printf(" %c | %c | %c\n",gridChar(b[0]),gridChar(b[1]),gridChar(b[2]));

printf("---+---+---\n");

printf(" %c | %c | %c\n",gridChar(b[3]),gridChar(b[4]),gridChar(b[5]));

printf("---+---+---\n");

printf(" %c | %c | %c\n",gridChar(b[6]),gridChar(b[7]),gridChar(b[8]));

}

int win(const int board[9]) {

//determines if a player has won, returns 0 otherwise.

unsigned wins[8][3] = {{0,1,2},{3,4,5},{6,7,8},{0,3,6},{1,4,7},{2,5,8},{0,4,8},{2,4,6}};

int i;

for(i = 0; i < 8; ++i) {

if(board[wins[i][0]] != 0 &&

board[wins[i][0]] == board[wins[i][1]] &&

board[wins[i][0]] == board[wins[i][2]])

return board[wins[i][2]];

}

return 0;

}

int minimax(int board[9], int player) {

//How is the position like for player (their turn) on board?

int winner = win(board);

if(winner != 0) return winner\*player;

int move = -1;

int score = -2;//Losing moves are preferred to no move

int i;

for(i = 0; i < 9; ++i) {//For all moves,

if(board[i] == 0) {//If legal,

board[i] = player;//Try the move

int thisScore = -minimax(board, player\*-1);

if(thisScore > score) {

score = thisScore;

move = i;

}//Pick the one that's worst for the opponent

board[i] = 0;//Reset board after try

}

}

if(move == -1) return 0;

return score;

}

void computerMove(int board[9]) {

int move = -1;

int score = -2;

int i;

for(i = 0; i < 9; ++i) {

if(board[i] == 0) {

board[i] = 1;

int tempScore = -minimax(board, -1);

board[i] = 0;

if(tempScore > score) {

score = tempScore;

move = i;

}

}

}

//returns a score based on minimax tree at a given node.

board[move] = 1;

}

void playerMove(int board[9]) {

int move = 0;

do {

printf("\nInput move ([1-9]): ");

scanf("%d", &move);

move--;

printf("\n");

} while (move >= 9 || move < 0 && board[move] == 0);

board[move] = -1;

}

int main() {

int board[9] = {0,0,0,0,0,0,0,0,0};

//computer squares are 1, player squares are -1.

printf("Computer: O , Player: X\nPlay 1st / 2nd ? ");

int player=0;

scanf("%d",&player);

printf("\n");

unsigned turn;

for(turn = 0; turn < 9 && win(board) == 0; ++turn) {

if((turn+player) % 2 == 0)

computerMove(board);

else {

draw(board);

playerMove(board);

}

}

switch(win(board)) {

case 0:

printf("Game Drawn. Better luck next time.\n");

break;

case 1:

draw(board);

printf("You lose, Sorry...\n");

break;

case -1:

printf("You win. Keep it up!\n");

break;

}

}

