**National Institute of Technology Silchar**

Project Title:

Tic Tac Toe in C language

Submitted by:

Rajib Das Shuva(14-15-088)

Mehedi Habib(14-15-086)

*Header Files Used:*

#include <stdio.h>

#include <stdlib.h>

*Global Variables & Functions:*

char matrix[5][5]; // 5X5 matrix

void init\_matrix(int); // initializing the matrix

void player\_move(); // player's move

void computer\_move\_3X3();

void computer\_move\_4X4();

void computer\_move\_5X5(); // computer's move

void disp\_matrix\_3X3();

void disp\_matrix\_4X4();

void disp\_matrix\_5X5(); // displaying the current matrix

char check\_3X3();

char check\_4X4();

char check\_5X5(); // checking if there is a winner

We try to explain all the user-defined functions used here one after another.

*void init\_matrix(int)*

Initially all indices are made empty.

int i,j;

for(i=0;i<n;i++)

for(j=0;j<n;j++)

matrix[i][j]=' ';

*void player\_move();*

According to the user input, we put a ‘X’ in that particular box. If the input is out of the matrix, we display an error message.

if(matrix[--x][--y]!=' '){ //if input index is not empty, invalid move is shown

printf("Invalid move, try again\n\n");

player\_move(); // makes the user input again

}

else

matrix[x][y]='X'; //else we put an ‘X’ in it

*void computer\_move\_3X3();*

After the 1st move from the user, now it is computer’s turn.

For its 1 st move, it searches a random empty index around the user’s input and puts a ‘O’ there.

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

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

if(matrix[i][j]=='X'){ // finds the user’s input

if(matrix[i][++j]==' '){ // searches for an empty index nearby

matrix[i][j]='O'; // puts a ‘O’ there

return;

}

else if(matrix[i][--j]==' '){

matrix[i][j]='O';

return;

}

else if(matrix[++i][j]==' '){

matrix[i][j]='O';

return;

}

else if(matrix[--i][j]==' '){

matrix[i][j]='O';

return;

}

}

}

}

Analyzing the user’s 2 nd and the following moves, computer wisely chooses its next moves. It first checks the rows to see whether it can win the match by putting an ‘O’ or prevents user from winning by blocking also with an ‘O’

// if 1st & 2nd elements are 'O' or 'X', we make 3rd 'O'

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

if((matrix[i][0]=='O'&& matrix[i][1]=='O')||(matrix[i][0]=='X'&& matrix[i][1]=='X')){ if(matrix[i][2]==' '){

matrix[i][2]='O';

return;

}

}

}

//if 1st & 3rd elements are 'O' or 'X', we make 2nd 'O'

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

if((matrix[i][0]=='O'&& matrix[i][2]=='O')||(matrix[i][0]=='X'&& matrix[i][2]=='X')){ if(matrix[i][1]==' '){

matrix[i][1]='O';

return;

}

}

}

//if 2nd & 3rd elements are 'O'or'X', we make 1st 'O'

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

if((matrix[i][1]=='O'&& matrix[i][2]=='O')||(matrix[i][1]=='X'&& matrix[i][2]=='X')){ if(matrix[i][0]==' '){

matrix[i][0]='O'; return;

}

}

}

Same goes for columns and diagonals too.

*void disp\_matrix\_3X3()*

We use the vertical bars (|) and dashes (---) to draw the lines and columns and try to give it a real rectangular shape like we draw while playing the game in the papers.

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

printf(" | | \n");

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

if(i!=2)

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

}

*char check\_3X3()*

We check the rows to see whether any of them are identical either with ‘X’ or ‘O’. If we find one, we simply return the ‘X’ or ‘O’.

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

if(matrix[i][0]==matrix[i][1]&&matrix[i][0]==matrix[i][2]) //checking if the elements of rows are identical

return matrix[i][0];

Same goes for columns and diagonals too.

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

if(matrix[0][i]==matrix[1][i]&&matrix[0][i]==matrix[2][i]) //checking if the elements of columns are identical

return matrix[0][i];

At this stage, there is no identical rows, columns or diagonals, we then check if there is any empty index left or not

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

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

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

return ' '; // if there, we return it }

}

return 'D'; // if not, the match is drawn

*int main()*

In the main function, we simply call every other user defined functions and check if there is a winner in every iteration by a do-while loop.

do{

player\_move(); //player's move

disp\_matrix\_3X3(); //displaying the current matrix

done=check\_3X3(); //checking the matrix

if(done!=' ')

break; //if the return is other than ' ',we break here

printf("\t Computer's move\n\n\n");

computer\_move\_3X3();

disp\_matrix\_3X3(); //displaying the matrix

done=check\_3X3(); //checking the matrix

} while(done==' ');