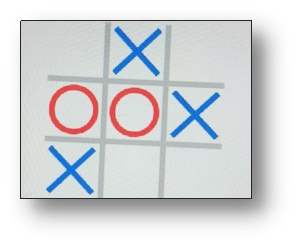
**NAME:-UPASNA**

**ROLL:-24csu225**

**ASSIGNMENT-6**

QUES- Tic Tac Toe Game

The working of tic tac toe game is same as traditional tic tac toe having following components:

Objective: To be the first to make a straight line with either 'X' or 'O'.

Game Board: The board consists of a 3x3 matrix-like structure, having 9 small boxes.

The computer: Since it is a two-player game each player gets one chance alternatively. i.e.; first player1 than player2

Moves: The computer starts the game with O. After that player makes moves alternatively.

Winning: You win by making your symbol in a row or diagonal or column. Also, as a part of strategy you need to block your opponent from forming a straight line while making of your own.

Sol:-

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#define PLAYER 'X'

#define COMPUTER 'O'

#define EMPTY ' '

void printBoard(char board[3][3]);

int checkWin(char board[3][3], char player);

int isBoardFull(char board[3][3]);

void playerMove(char board[3][3]);

void computerMove(char board[3][3]);

int main() {

char board[3][3] = {

{EMPTY, EMPTY, EMPTY},

{EMPTY, EMPTY, EMPTY},

{EMPTY, EMPTY, EMPTY}

};

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

printBoard(board);

while (1) {

playerMove(board);

printBoard(board);

if (checkWin(board, PLAYER)) {

printf("Player wins!\n");

break;

}

if (isBoardFull(board)) {

printf("It's a draw!\n");

break;

}

computerMove(board);

printBoard(board);

if (checkWin(board, COMPUTER)) {

printf("Computer wins!\n");

break;

}

if (isBoardFull(board)) {

printf("It's a draw!\n");

break;

}

}

return 0;

}

void printBoard(char board[3][3]) {

printf("\n");

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

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

printf(" %c ", board[i][j]);

if (j < 2) printf("|");

}

printf("\n");

if (i < 2) printf("---|---|---\n");

}

printf("\n");

}

int checkWin(char board[3][3], char player) {

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

// Check rows and columns

if ((board[i][0] == player && board[i][1] == player && board[i][2] == player) ||

(board[0][i] == player && board[1][i] == player && board[2][i] == player)) {

return 1;

}

}

// Check diagonals

if ((board[0][0] == player && board[1][1] == player && board[2][2] == player) ||

(board[0][2] == player && board[1][1] == player && board[2][0] == player)) {

return 1;

}

return 0;

}

int isBoardFull(char board[3][3]) {

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

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

if (board[i][j] == EMPTY) return 0;

return 1;

}

void playerMove(char board[3][3]) {

int row, col;

while (1) {

printf("Enter your move (row and column: 0, 1, or 2): ");

scanf("%d %d", &row, &col);

if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == EMPTY) {

board[row][col] = PLAYER;

break;

} else {

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

}

}

}

void computerMove(char board[3][3]) {

srand(time(0));

int row, col;

while (1) {

row = rand() % 3;

col = rand() % 3;

if (board[row][col] == EMPTY) {

board[row][col] = COMPUTER;

break;

       }

    }

}