```
//Tic Tac Toe


#include<bits/stdc++.h>
using namespace std;

#define COMPUTER 1
#define HUMAN 2

#define SIDE 3

#define COMPUTERMOVE 'O'
#define HUMANMOVE 'X'


void showBoard(char board[][SIDE])
{

	printf("\t\t\t %c | %c | %c \n", board[0][0], board[0][1], board[0][2]);
	printf("\t\t\t-----------\n");
	printf("\t\t\t %c | %c | %c \n", board[1][0], board[1][1], board[1][2]);
	printf("\t\t\t-----------\n");
	printf("\t\t\t %c | %c | %c \n\n", board[2][0], board[2][1], board[2][2]);
}


void showInstructions()
{
	printf("\nChoose a cell numbered from 1 to 9 as below and play\n\n");

	printf("\t\t\t 1 | 2 | 3 \n");
	printf("\t\t\t-----------\n");
	printf("\t\t\t 4 | 5 | 6 \n");
	printf("\t\t\t-----------\n");
	printf("\t\t\t 7 | 8 | 9 \n\n");
}


void initialise(char board[][SIDE])
{
	// Initially the board to '*' as said
	for (int i=0; i<SIDE; i++)
	{
		for (int j=0; j<SIDE; j++)
```

```
                        board[i][j] = '*';
        }
}

void declareWinner(int whoseTurn)
{
        if (whoseTurn == COMPUTER)
                printf("COMPUTER has won\n");
        else
                printf("HUMAN has won\n");
}

bool rowCrossed(char board[][SIDE])
{
        for (int i=0; i<SIDE; i++)
        {
                if (board[i][0] == board[i][1] &&
                        board[i][1] == board[i][2] &&
                        board[i][0] != '*')
                        return (true);
        }
        return(false);
}

bool columnCrossed(char board[][SIDE])
{
        for (int i=0; i<SIDE; i++)
        {
                if (board[0][i] == board[1][i] &&
                        board[1][i] == board[2][i] &&
                        board[0][i] != '*')
                        return (true);
        }
        return(false);
}

bool diagonalCrossed(char board[][SIDE])
{
        if (board[0][0] == board[1][1] &&
                board[1][1] == board[2][2] &&
                board[0][0] != '*')
                return(true);

        if (board[0][2] == board[1][1] &&
```

```
                board[1][1] == board[2][0] &&
                board[0][2] != '*')
                return(true);

        return(false);
}


bool gameOver(char board[][SIDE])
{
        return(rowCrossed(board) || columnCrossed(board) || diagonalCrossed(board) );
}


int minimax(char board[][SIDE], int depth, bool isAI)
{
        int score = 0;
        int bestScore = 0;
        if (gameOver(board) == true)
        {
                if (isAI == true)
                        return -10;
                if (isAI == false)
                        return +10;
        }
        else
        {
                if(depth < 9)
                {
                        if(isAI == true)
                        {
                                bestScore = -999;
                                for(int i=0; i<SIDE; i++)
                                {
                                        for(int j=0; j<SIDE; j++)
                                        {
                                                if (board[i][j] == '*')
                                                {
                                                        board[i][j] = COMPUTERMOVE;
                                                        score = minimax(board, depth + 1, false);
                                                        board[i][j] = '*';
                                                        if(score > bestScore)
                                                        {
                                                                bestScore = score;
```

```
                                        }
                                    }
                                }
                            }
                            return bestScore;
                        }
                        else
                        {
                            bestScore = 999;
                            for (int i = 0; i < SIDE; i++)
                            {
                                for (int j = 0; j < SIDE; j++)
                                {
                                    if (board[i][j] == '*')
                                    {
                                        board[i][j] = HUMANMOVE;
                                        score = minimax(board, depth + 1, true);
                                        board[i][j] = '*';
                                        if (score < bestScore)
                                        {
                                            bestScore = score;
                                        }
                                    }
                                }
                            }
                            return bestScore;
                        }
                    }
                    else
                    {
                        return 0;
                    }
                }
            }

            int bestMove(char board[][SIDE], int moveIndex)
            {
                int x = -1, y = -1;
                int score = 0, bestScore = -999;
                for (int i = 0; i < SIDE; i++)
                {
                    for (int j = 0; j < SIDE; j++)
                    {
                        if (board[i][j] == '*')
```

```
                        {
                                board[i][j] = COMPUTERMOVE;
                                score = minimax(board, moveIndex+1, false);
                                board[i][j] = '*';
                                if(score > bestScore)
                                {
                                        bestScore = score;
                                        x = i;
                                        y = j;
                                }
                        }
                }
        }
        return x*3+y;
}

// A function to play Tic-Tac-Toe
void playTicTacToe(int whoseTurn)
{
        char board[SIDE][SIDE];
        int moveIndex = 0, x = 0, y = 0;

        initialise(board);
        showInstructions();


        while (gameOver(board) == false && moveIndex != SIDE*SIDE)
        {
                int n;
                if (whoseTurn == COMPUTER)
                {
                        n = bestMove(board, moveIndex);
                        x = n / SIDE;
                        y = n % SIDE;
                        board[x][y] = COMPUTERMOVE;
                        printf("COMPUTER has put a %c in cell %d\n\n", COMPUTERMOVE,
n+1);
                        showBoard(board);
                        moveIndex ++;
                        whoseTurn = HUMAN;
                }

                else if (whoseTurn == HUMAN)
                {
```

```
                        printf("You can insert in the following positions : ");
                        for(int i=0; i<SIDE; i++)
                                for (int j = 0; j < SIDE; j++)
                                        if (board[i][j] == '*')
                                                printf("%d ", (i * 3 + j) + 1);
                        printf("\n\nEnter the position = ");
                        scanf("%d",&n);
                        n--;
                        x = n / SIDE;
                        y = n % SIDE;
                        if(board[x][y] == '*' && n<9 && n>=0)
                        {
                                board[x][y] = HUMANMOVE;
                                printf ("\nHUMAN has put a %c in cell %d\n\n", HUMANMOVE,
n+1);
                                showBoard(board);
                                moveIndex ++;
                                whoseTurn = COMPUTER;
                        }
                        else if(board[x][y] != '*' && n<9 && n>=0)
                        {
                                printf("\nPosition is occupied, select any one place from the
available places\n\n");
                        }
                        else if(n<0 || n>8)
                        {
                                printf("Invalid position\n");
                        }
                }
        }

        if (gameOver(board) == false && moveIndex == SIDE * SIDE)
                printf("It's a draw\n");
        else
        {
                if (whoseTurn == COMPUTER)
                        whoseTurn = HUMAN;
                else if (whoseTurn == HUMAN)
                        whoseTurn = COMPUTER;

                declareWinner(whoseTurn);
        }
}
```

```
int main()
{
	printf("\n-----------------------------------------------------------------\n\n");
	printf("\t\t\t Tic-Tac-Toe\n");
	printf("\n-----------------------------------------------------------------\n\n");
	char cont='y';
	do {
		char choice;
		printf("Do you want to start first?(y/n) : ");
		scanf(" %c", &choice);

		if(choice=='n')
			playTicTacToe(COMPUTER);
		else if(choice=='y')
			playTicTacToe(HUMAN);
		else
			printf("Invalid choice\n");

		printf("\nDo you want to quit(y/n) : ");
		scanf(" %c", &cont);
	} while(cont=='n');
	return (0);
}
```