Exercise 6: Implementation of min-max algorithm for an application

Min-Max Algorithm:

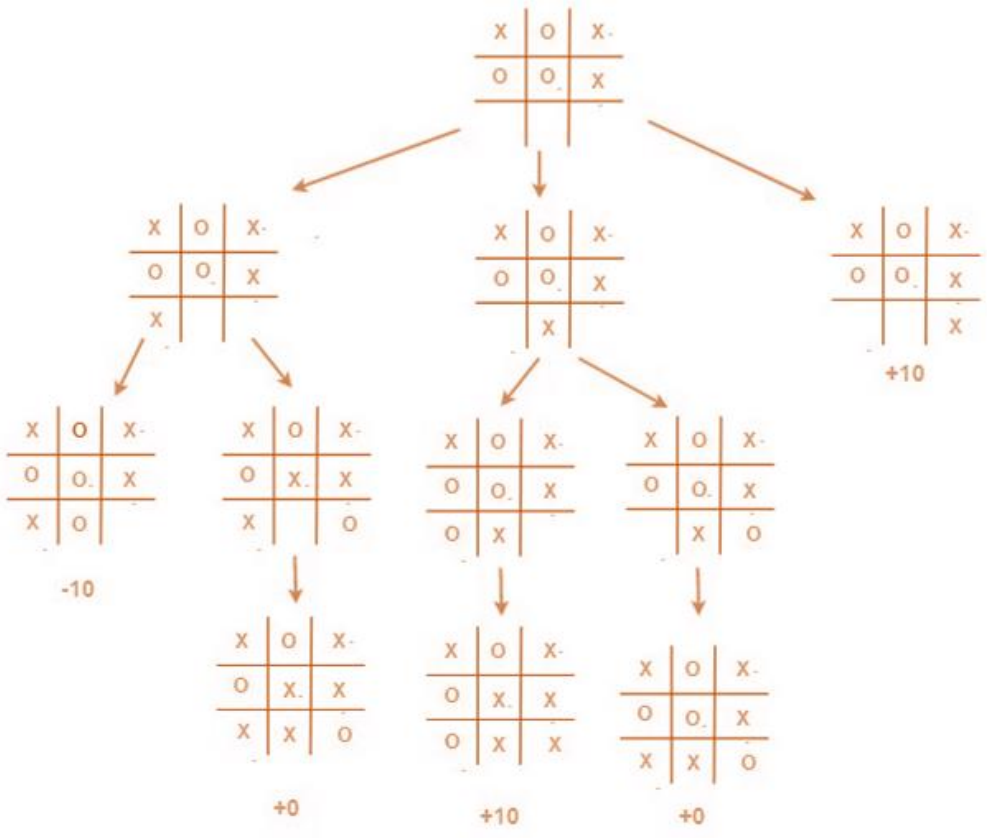
Aim:

To find the next best move in Tic Tac Toe game using Min Max Algorithm.

Algorithm:

In **Min-max** the two players are called maximizer and minimizer. The maximizer tries to get the highest score possible while the minimizer tries to do the opposite and get the lowest score possible.

1. Maximizer goes LEFT: It is now the minimizers turn.
   Minimizer now has a choice between 3 and 5. Being the minimizer it will definitely choose the least among both, that is 3.
2. Maximizer goes RIGHT: It is now the minimizers turn.
   Minimizer now has a choice between 2 and 9. It will choose 2 as it is the least among the two values.



Program:

```
#include<bits/stdc++.h>

using namespace std;
```

```
struct Move
{
	int row, col;
};

char player = 'x', opponent = 'o';
bool isMovesLeft(char board[3][3])
{
	for (int i = 0; i<3; i++)
		for (int j = 0; j<3; j++)
			if (board[i][j]=='_')
				return true;
	return false;
}

int evaluate(char b[3][3])
{

	for (int row = 0; row<3; row++)
	{
		if (b[row][0]==b[row][1] &&
			b[row][1]==b[row][2])
		{
			if (b[row][0]==player)
				return +10;
			else if (b[row][0]==opponent)
				return -10;
		}
	}
```

```
for (int col = 0; col<3; col++)
{
        if (b[0][col]==b[1][col] &&
                b[1][col]==b[2][col])
        {
                if (b[0][col]==player)
                        return +10;

                else if (b[0][col]==opponent)
                        return -10;
        }
}


if (b[0][0]==b[1][1] && b[1][1]==b[2][2])
{
        if (b[0][0]==player)
                return +10;
        else if (b[0][0]==opponent)
                return -10;
}

if (b[0][2]==b[1][1] && b[1][1]==b[2][0])
{
        if (b[0][2]==player)
                return +10;
        else if (b[0][2]==opponent)
                return -10;
}
```

```
	return 0;
}


int minimax(char board[3][3], int depth, bool isMax)
{
	int score = evaluate(board);


	if (score == 10)
		return score;
	if (score == -10)
		return score;

	if (isMovesLeft(board)==false)
		return 0;


	if (isMax)
	{
		int best = -1000;


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

				if (board[i][j]=='_')
				{
```

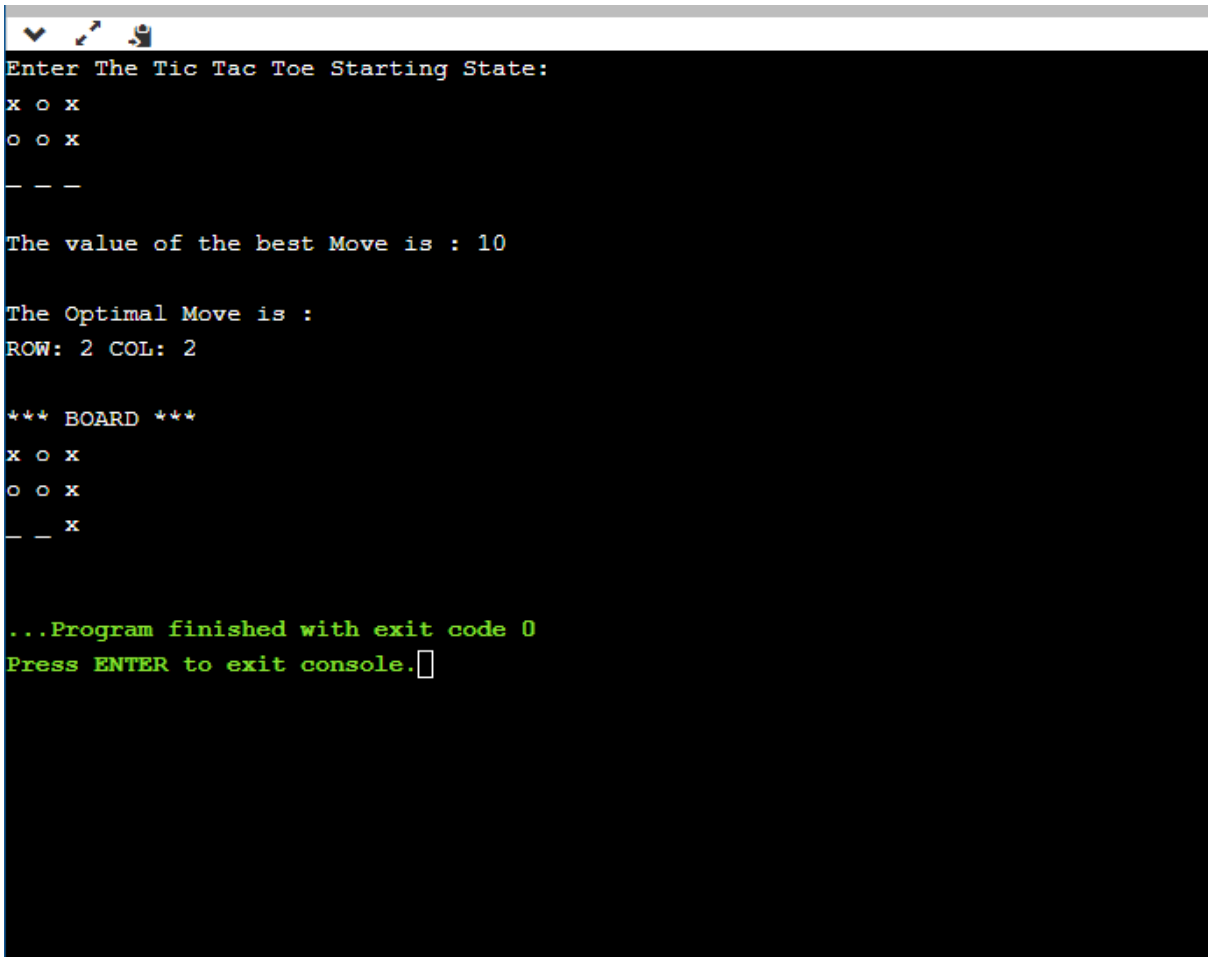
```
                                board[i][j] = player;



                                best = max( best,
                                        minimax(board, depth+1, !isMax) );

                                board[i][j] = '_';
                        }
                }
        }
        return best;
}



else
{
        int best = 1000;



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

                        if (board[i][j]=='_')
                        {

                                board[i][j] = opponent;
```

```
                                        best = min(best,
                                                minimax(board, depth+1, !isMax));



                                        board[i][j] = '_';
                                }
                        }
                }
                return best;
        }
}


Move findBestMove(char board[3][3])
{
        int bestVal = -1000;
        Move bestMove;
        bestMove.row = -1;
        bestMove.col = -1;


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

                        if (board[i][j]=='_')
                        {

                                board[i][j] = player;
```

```
                                int moveVal = minimax(board, 0, false);

                                board[i][j] = '_';

                                if (moveVal > bestVal)
                                {
                                        bestMove.row = i;
                                        bestMove.col = j;
                                        bestVal = moveVal;
                                }

                        }
                }
        }



        cout<<"\nThe value of the best Move is : "<<bestVal<<"\n\n";

        return bestMove;
}

int main()
{
        char board[3][3];
   cout<<"Enter The Tic Tac Toe Starting State:\n";
        for (int i = 0; i<3; i++)
                {
                        for (int j = 0; j<3; j++)
                        {
```

```
                cin>>board[i][j];
            }
        }

    Move bestMove = findBestMove(board);

    printf("The Optimal Move is :\n");
    printf("ROW: %d COL: %d\n\n", bestMove.row,bestMove.col );

    for (int i = 0; i<3; i++)
        {
            for (int j = 0; j<3; j++)
            {
                if(i==bestMove.row && j==bestMove.col)
                board[i][j]='x';
            }
        }

        cout<<"*** BOARD ***\n";
        for (int i = 0; i<3; i++)
        {
            for (int j = 0; j<3; j++)
            {
                cout<<board[i][j]<<" ";
            }
            cout<<"\n";
        }
    return 0;
}
```

OUTPUT:



Result:

The given program was successfully created using Min-Max Algorithm and was successfully executed in an Online C++ compiler.