IMPLEMENTATION OF MINIMAX ALGORITHM FOR AN APPLICATION

NAME: **RISHAL RAMESH** ***EXP NO:*** **6**

REG NO: **RA1911030010084**

AIM:

To implement min max algorithm as a tic tac toe game.

DESCRIPTION:

Minimax is a kind of backtracking algorithm that is used in decision making and game theory to find the optimal move for a player, assuming that your opponent also plays optimally. It is widely used in two player turn-based games such as Tic-Tac-Toe, Backgammon, Mancala, Chess, etc.
In Minimax 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.

CODE:

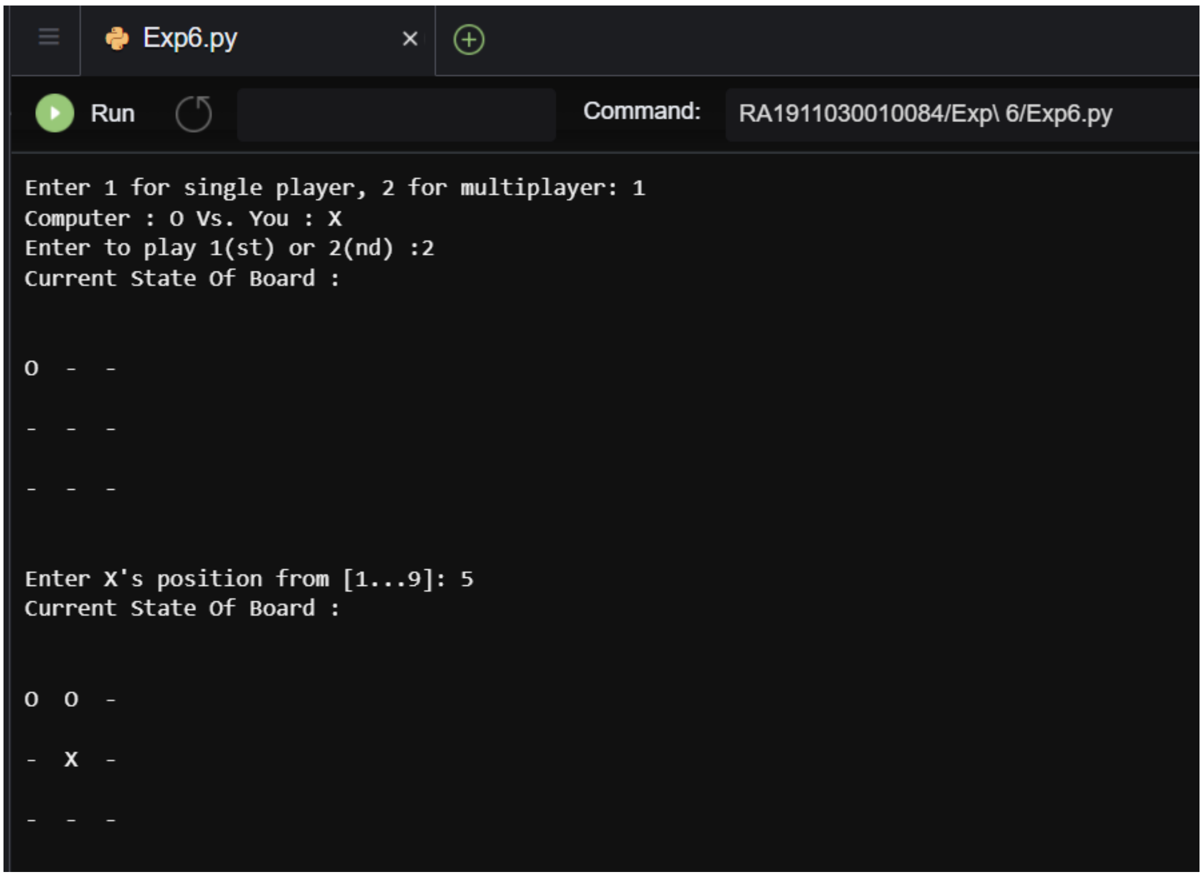
```
def ConstBoard(board):
    print("Current State Of Board : \n\n");
    for i in range (0,9):
        if((i>0) and (i%3)==0):
            print("\n");
        if(board[i]==0):
            print("- ",end=" ");
        if (board[i]==1):
            print("O ",end=" ");
        if(board[i]==-1):
            print("X ",end=" ");
    print("\n\n");


#This function takes the user move as input and make the required changes on the board.
def User1Turn(board):
    pos=input("Enter X's position from [1...9]: ");
    pos=int(pos);
    if(board[pos-1]!=0):
        print("Wrong Move!!!");
        exit(0) ;
    board[pos-1]=-1;


def User2Turn(board):
    pos=input("Enter O's position from [1...9]: ");
    pos=int(pos);
    if(board[pos-1]!=0):
        print("Wrong Move!!!");
        exit(0);
    board[pos-1]=1;


#MinMax function.
def minimax(board,player):
    x=analyzeboard(board);
    if(x!=0):
        return (x*player);
    pos=-1;
    value=-2;
    for i in range(0,9):
        if(board[i]==0):
            board[i]=player;
            score=-minimax(board,(player*-1));
            if(score>value):
                value=score;
                pos=i;
            board[i]=0;

    if(pos==-1):
        return 0;
    return value;

#This function makes the computer's move using minmax algorithm.
def CompTurn(board):
    pos=-1;
    value=-2;
    for i in range(0,9):
        if(board[i]==0):
            board[i]=1;
            score=-minimax(board, -1);
            board[i]=0;
            if(score>value):
                value=score;
                pos=i;

    board[pos]=1;
```

```
#This function is used to analyze a game.
def analyzeboard(board):
    cb=[[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]];

    for i in range(0,8):
        if(board[cb[i][0]] != 0 and
           board[cb[i][0]] == board[cb[i][1]] and
           board[cb[i][0]] == board[cb[i][2]]):
            return board[cb[i][2]];
    return 0;

#Main Function.
def main():
    choice=input("Enter 1 for single player, 2 for multiplayer: ");
    choice=int(choice);
    #The broad is considered in the form of a single dimentional array.
    #One player moves 1 and other move -1.
    board=[0,0,0,0,0,0,0,0,0];
    if(choice==1):
        print("Computer : O Vs. You : X");
        player= input("Enter to play 1(st) or 2(nd) :");
        player = int(player);
        for i in range (0,9):
            if(analyzeboard(board)!=0):
                break;
            if((i+player)%2==0):
                CompTurn(board);
            else:
                ConstBoard(board);
                User1Turn(board);
    else:
        for i in range (0,9):
            if(analyzeboard(board)!=0):
                break;
            if((i)%2==0):
                ConstBoard(board);
                User1Turn(board);
            else:
                ConstBoard(board);
                User2Turn(board);



    x=analyzeboard(board);
    if(x==0):
         ConstBoard(board);
         print("Draw!!!")
    if(x==-1):
         ConstBoard(board);
         print("X Wins!!! Y Loose !!!")
    if(x==1):
         ConstBoard(board);
         print("X Loose!!! O Wins !!!!")

#---------------#
```

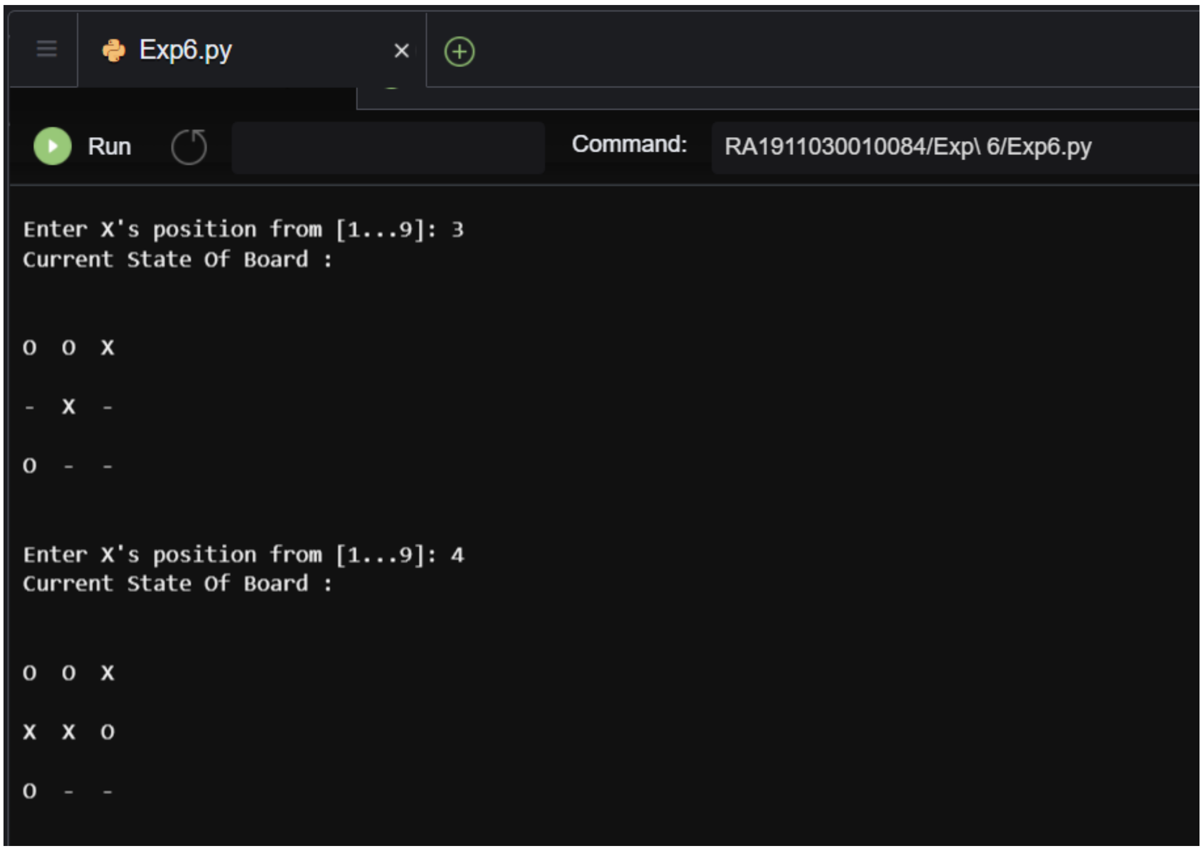
main()

#--------------#

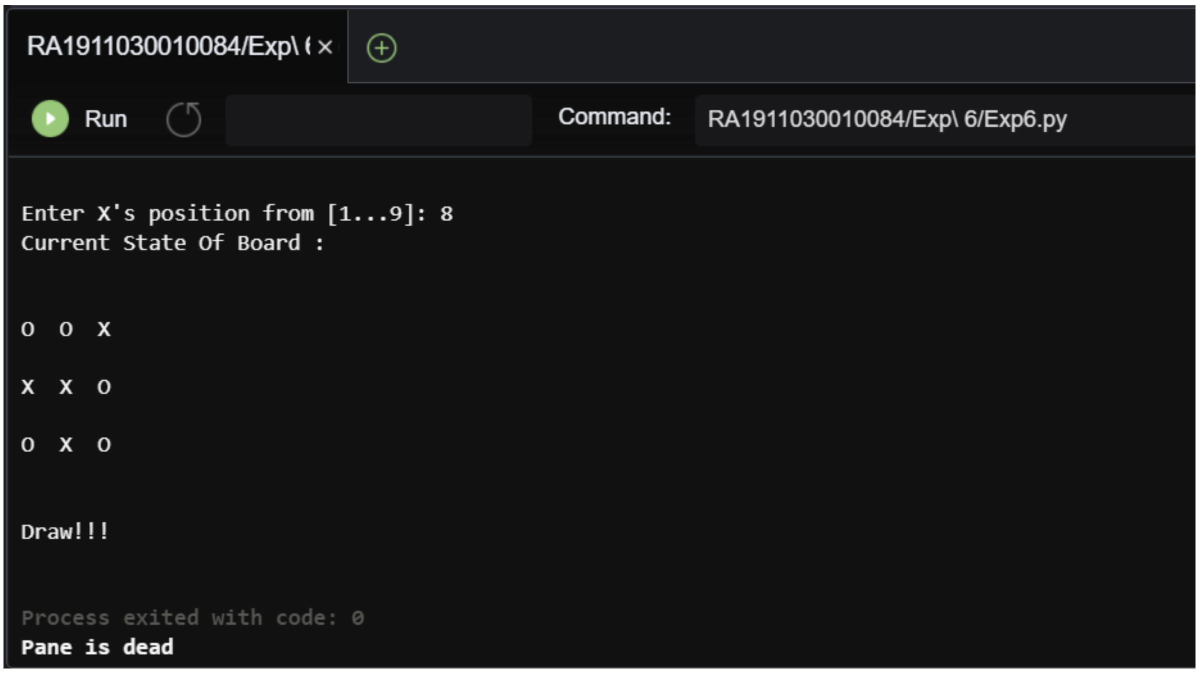
OUTPUT:



```
Enter 1 for single player, 2 for multiplayer: 1
Computer : O Vs. You : X
Enter to play 1(st) or 2(nd) :2
Current State Of Board :


O  -  -

-  -  -

-  -  -


Enter X's position from [1...9]: 5
Current State Of Board :


O  O  -

-  X  -

-  -  -
```



```
Enter X's position from [1...9]: 3
Current State Of Board :


O  O  X

-  X  -

O  -  -


Enter X's position from [1...9]: 4
Current State Of Board :


O  O  X

X  X  O

O  -  -
```



```
Enter X's position from [1...9]: 8
Current State Of Board :


O  O  X

X  X  O

O  X  O


Draw!!!


Process exited with code: 0
Pane is dead
```

RESULT:

Min-max algorithm was successfully implemented in python.