EX NO: 5 MINIMAX ALGORITHM

Date:

AIM:

To Implement Minimax algorithm Problem usi

Python.

Algorithm:

* Initialize the Board. Create a 3x3 board

assign Players.

* use the minimax algorithm to evaluate Po

moves and select the optimal one for

computer

Alternate between human and computer turns,

updating the board and checking for a wi

draw after each move

Display the result.

rogram:

```
rom Math import inf as infinity from ra
port choice
port Platform
port time
m us importent system
noxan = -1
mP = +1
rd= [[0,0,0], [0,0,0],]
evaluate (state):
```

```
    if wins [state, comp]:
        score += 1
    elif wins (state, HUMAN)
        score = 0
    return score
def wins (state, player):
win_state = [[state [0], [0], state [0] [1],
  state [0] [2]], [state [1] [0], state [1] [1],
  state [2], [2]]
[state [2] [0], state [2], [1], state [2] [2]]
[state [0] [1], state [1] [1], state [2] [1]]
[state [2] [0], state [1] [1], state [0] [2]]
if [player, player, player] in win_state:
    return True
else
    return flase
def game_over(state)
  return wins (state, Human) or wins (sGate, Com
def empty_cells (state):
cells = []
for x, row in enumerate (state):
  for y, cell in enumerate(row):
      if cell = 0:
          cells.append ([x,y])
return cells
```

```
def valid - move (x, y):
    if [x, y] in empty - cells (board):
        return True

    else:
        return False
def set - move (x, y, Player):
    if valid. move(x, y):
        board [x] [y] = Player
        return True
    else
        return False
def minimax (state, depth, Player):
    if Player = = comP;
        best = [-1, -1, - infinity]
    else:
        best = [-1, -1, + infinity]
if depth = = 0 or game - over (state):
        score = evaluate (state)
        return [-1, -1, score]
for cell in empty - cells (state):
        x, y = cell [0], cell [1]
    state [x] [y] = Player
score = minimax (state, depth -1, - Player)
  state [x] [y] = 0
score [0], score [1] = x, y
if Player = = comP:
    if score [2] > best [2];
best = score
```

19/11/2024 17:54

```
else
   if score [2] < best [2];
   best = score
return best
def ai_turn (c-choice, b-choice):
depth = len (empty_cells (board)
if depth ==0 len game_over (board);
return
def human_turn (c-choice, h-choice):
   depth = len (empty -cells (board)
if depth ==0 len game- over (board):
return
move= -1
mover = {
   1= [0,0] , 2 [0,1], 3: [0,2]
   4: [1,0]  5: [1,1]  6: [1,2]
   7: [2,0]  8: [2,1]  9: [2,2]
}
def main():
   clean()
   h-choice = ''
   c - choice = ''
   first = ''
while h-choice != 'o' and h-choice != 'x':
try:
 print('')
h - choice= input ['choice x or o\n chose:'] upper')
except [EofError, keyboard Interrupt]:
print ('Bye')
```

```
exit()
except(KeyError, V-ValueError):
    Print('Bad Choice')
if h-choice == 'x'
    c-choice = 'o'
else
    c-choice = 'x'
    clean()
if_name_ == '_main_':
    main()
```

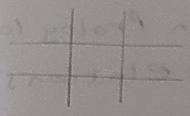
output;

Choice x or o

Choice: x

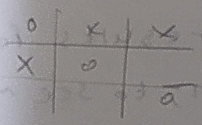
First to start? [y/n] = Y

Human turn [x]



use numpad (1...9): 4

Completed turn [o]



You lose!

Result:

Thus the Program has been successfully executed.