EXNO: 05
DATE:

MINIMAX ALGORITHM.

AIM:-

Create a program to solve MinMax Algorithm problem using python code.

CODE:-

```
import math
Player-X = "X"
Player-O = "O"
Empty = " "
def print_board(board):
    for row in board:
        print("|".join(row))
        print("-" * 9)
def check_winner(board):
    for row in board:
        if row[0] == row[1] == row[2] != Empty:
            return row[0]
    for col in range(3):
        if board[0][col] == board[1][col] ==
                board[2][col] != Empty:
            return board[0][1]
    if board[0][0] == board[1][1] == board[2][2] != Empty:
        return board[0][0]
    if board[0][2] == board[1][1] == board[2][0] != empty
        return board[0][2]
    return None
def is_full(board):
    for row in board:
        if empty in row:
            return false
    return true.
```

```
# Minimax function :-
def minimax (board, depth, is_maximizing):
    winner = check_winner (board)
    if winner == Player_X :
        return 1
    elif winner == Player_O:
        return -1
    elif is_full(board):
        return 0
    if is_maximizing :
        best_score = - math.inf
        for i in range(3):
            for j in range (3):
                if board [i] [j] == Empty :
                    board [i] [j] = Player_X
                    score = minimax (board, depth+1, False)
                    board [i][j] = Empty.
                best_score = min (score, best_score)
        return best_score
def find_best_move (board):
    best_move = - math.inf
    best_move = none
    for i in range (3):
        for j in range (3):
            if board [i] [j] == Empty :
                board [i] [j] = Player_X
                score = minimax (board, 0, False)
                board [i][j] = Empty
                if score > best_score:
                best_score = score
                best_move = (i, j)
        return best_move
board = [[Player_X, Player_O, Player_X],
         [Player_O, Player_X, Empty],
         [empty, Empty, Player O]]
```

```
print("Initial board:")
print_board(board)
best_move = find_best_move(board)
if best_move:
    board[best_move[0]][best_move[1]] = player_x
    print("\nBoard after player_x best move:")
    print_board(board)
else:
    print("no move left).
```

Output:-

```
Initial board:
X | O | X
---------
O | X |
---------
  |   | O
---------

Board after PLAYER_X's best move:
X | O | X
---------
O | X |
---------
X |   | O
---------
```

Result:-

Thus Minmax Algorithm is executed and output is verified.