KJSCE/IT/SY/SEM IV/HO-IAI/2022-23

**Batch:B4 Experiment Number:4**

**Roll Number:16010421119 Name: Aarya Tiwari**

**Aim of the Experiment:** Implementation of Adversarial algorithm-Min-Max for Tic-Tac-Toe Game

**Program/ Steps:**

-> import math

def print\_board(board):

print('-------------')

for i in range(3):

print('|', end=' ')

for j in range(3):

print(board[i\*3+j], '|', end=' ')

print()

print('-------------')

def evaluate(board):

# Check for rows

for i in range(0, 9, 3):

if board[i] == board[i+1] == board[i+2]:

if board[i] == 'X':

return 10

elif board[i] == 'O':

return -10

# Check for columns

for i in range(3):

if board[i] == board[i+3] == board[i+6]:

if board[i] == 'X':

return 10

elif board[i] == 'O':

return -10

# Check for diagonal

if board[0] == board[4] == board[8]:

if board[0] == 'X':

return 10

elif board[0] == 'O':

return -10

if board[2] == board[4] == board[6]:

if board[2] == 'X':

return 10

elif board[2] == 'O':

return -10

return 0

def minimax(board, depth, is\_maximizing):

score = evaluate(board)

if score == 10:

return score - depth

if score == -10:

return score + depth

if '-' not in board:

return 0

if is\_maximizing:

best\_score = -math.inf

for i in range(9):

if board[i] == '-':

board[i] = 'X'

score = minimax(board, depth+1, False)

board[i] = '-'

best\_score = max(score, best\_score)

return best\_score

else:

best\_score = math.inf

for i in range(9):

if board[i] == '-':

board[i] = 'O'

score = minimax(board, depth+1, True)

board[i] = '-'

best\_score = min(score, best\_score)

return best\_score

def get\_best\_move(board):

best\_score = -math.inf

best\_move = None

for i in range(9):

if board[i] == '-':

board[i] = 'X'

score = minimax(board, 0, False)

board[i] = '-'

if score > best\_score:

best\_score = score

best\_move = i

return best\_move

board = ['-' for i in range(9)]

print\_board(board)

while True:

human\_move = int(input('Enter your move (0-8): '))

board[human\_move] = 'O'

print\_board(board)

if evaluate(board) == -10:

print('HuPlayer win!')

break

if '-' not in board:

print('Tie!')

break

computer\_move = get\_best\_move(board)

board[computer\_move] = 'X'

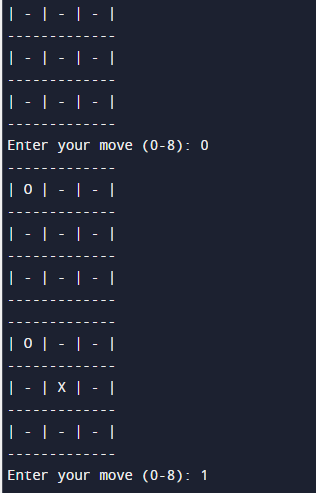
print\_board(board)

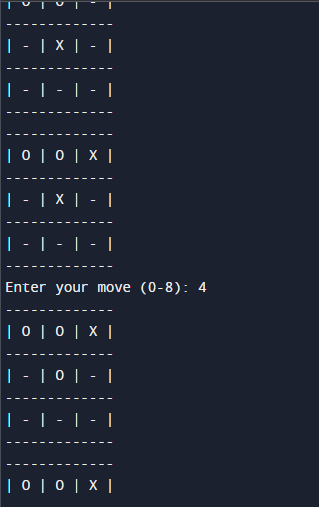
if evaluate(board) == 10:

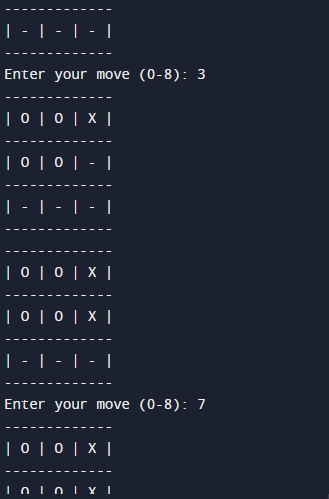
print('HuPlayer lose!')

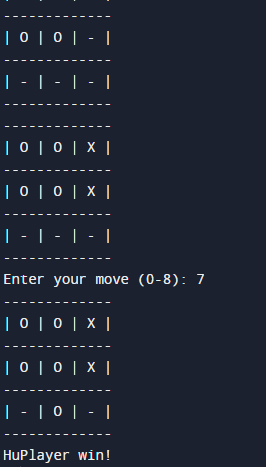
break

**Output/Result:**

****

****

****

****

**Post Lab Question-Answers:**

# Game playing is often called as an

* 1. Non-adversial search
  2. **Adversial search**
  3. Sequential search
  4. None of the above

# What are the basic requirements or need of AI search methods in game playing?

* 1. **Initial State of the game**
  2. **Operators defining legal moves**
  3. **Successor functions**
  4. d) Goal test
  5. e) Path cost

# Outcomes:

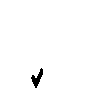
# CO2:Analyze and formalize the problem (as a state space, graph, etc.) and select

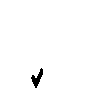
# the appropriatesearch method and write the algorithm

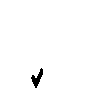
**Conclusion (based on the Results and outcomes achieved):**

By conducting this experiment we Implementated Adversarial algorithm-Min-Max for Tic-Tac-Toe Game using python and the required output was received.

# References:

 How to make your Tic Tac Toe game unbeatable by using the minimax algorithm: [https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-](https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm-9d690bad4b37/#%3A~%3Atext%3DA%20Minimax%20algorithm%20can%20be%2Con%20each%20available%20spot%20(recursion)) [by-using-the-minimax-algorithm-](https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm-9d690bad4b37/#%3A~%3Atext%3DA%20Minimax%20algorithm%20can%20be%2Con%20each%20available%20spot%20(recursion)) [9d690bad4b37/#:~:text=A%20Minimax%20algorithm%20can%20be,on%20each%20a](https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm-9d690bad4b37/#%3A~%3Atext%3DA%20Minimax%20algorithm%20can%20be%2Con%20each%20available%20spot%20(recursion)) [vailable%20spot%20(recursion)](https://www.freecodecamp.org/news/how-to-make-your-tic-tac-toe-game-unbeatable-by-using-the-minimax-algorithm-9d690bad4b37/#%3A~%3Atext%3DA%20Minimax%20algorithm%20can%20be%2Con%20each%20available%20spot%20(recursion))

** Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, 2ndEdition, Pearson Publication**

** Elaine Rich, Kevin Knight, Artificial Intelligence, Tata McGraw Hill, 1999.**

(A Constituent College of Somaiya Vidyavihar University)