# Bahria University,

## Karachi Campus



LAB EXPERIMENT NO.

**\_05\_**

LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| 01 | Implement Tic Tac Toe game by using Min Max Algorithm (Adversial Search) which suggest user a best move. |
|  |  |
|  |  |
|  |  |
|  |  |

Submitted On:

Date: 02/04/2024

**Task No 01:** Implement Tic Tac Toe game by using Min Max Algorithm (Adversial Search) which suggest user a best move.

**Solution:**

import math

board = [" " for \_ in range(9)]

moves\_array = []  # Array to store moves for analysis

def print\_board():

for i in range(3):

print("| " + " | ".join(board[i\*3:(i+1)\*3]) + " |")

def available\_moves():

return [i for i, x in enumerate(board) if x == " "]

def make\_move(position, player):

board[position] = player

moves\_array.append(position)  # Store the move in the array

def is\_winner(player):

for i in range(3):

if board[i\*3] == board[i\*3+1] == board[i\*3+2] == player: return True

if board[i] == board[i+3] == board[i+6] == player: return True

if board[0] == board[4] == board[8] == player: return True

if board[2] == board[4] == board[6] == player: return True

return False

def minimax(is\_maximizing, player):

if is\_winner('X'):

return 1

elif is\_winner('O'):

return -1

elif len(available\_moves()) == 0:

return 0

if is\_maximizing:

best\_score = -math.inf

for move in available\_moves():

make\_move(move, player)

score = minimax(False, 'O' if player == 'X' else 'X')

board[move] = " "

moves\_array.pop()  # Remove the move from the array

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

return best\_score

else:

best\_score = math.inf

for move in available\_moves():

make\_move(move, player)

score = minimax(True, 'O' if player == 'X' else 'X')

board[move] = " "

moves\_array.pop()  # Remove the move from the array

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

return best\_score

def best\_move(player):

best\_score = -math.inf if player == 'X' else math.inf

move = -1

for i in available\_moves():

make\_move(i, player)

score = minimax(player == 'O', 'O' if player == 'X' else 'X')

board[i] = " "

moves\_array.pop()  # Remove the move from the array

if (player == 'X' and score > best\_score) or (player == 'O' and score < best\_score):

best\_score = score

move = i

return move

def human\_hint():

move = best\_move('O')

print(f"Hint: The best move for you is at position {move}")

def human\_move():

human\_hint()  # Call the hint function

move = -1

while move not in available\_moves():

try:

move = int(input("Enter your move (0-8): "))

if move not in available\_moves():

print("Invalid move. Try again.")

except ValueError:

print("Invalid input. Please enter a number.")

make\_move(move, 'O')

def is\_draw():

return len(available\_moves()) == 0 and not is\_winner('X') and not is\_winner('O')

def play\_game():

current\_player = 'O'  # Human goes first

while not is\_winner('X') and not is\_winner('O') and not is\_draw():

print\_board()

if current\_player == 'O':

human\_move()

else:

print("Computer's turn:")

move = best\_move('X')

make\_move(move, 'X')

print(f"Computer chose position {move}")

current\_player = 'X' if current\_player == 'O' else 'O'

print\_board()

if is\_winner('X'):

print("Computer wins!")

elif is\_winner('O'):

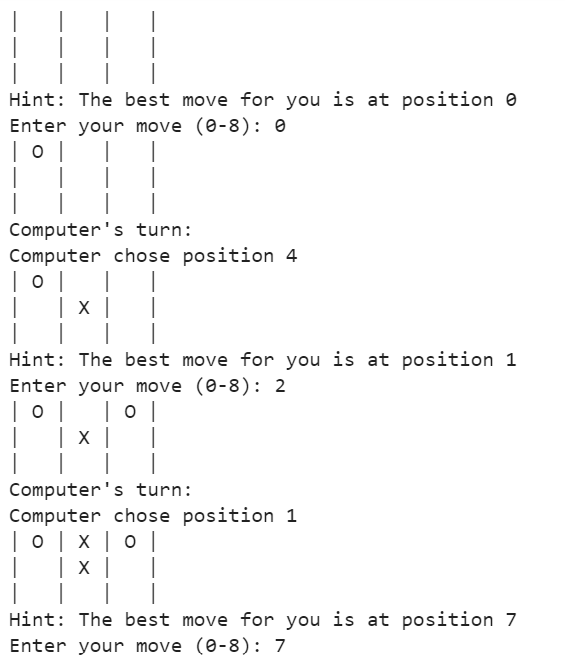
print("You win!")

else:

print("It's a draw!")

play\_game()

**Output:**

****

**A screenshot of a computer program

Description automatically generated**