# Bahria University,

# A logo with text on it Description automatically generatedKARACHI 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: 25/2/2023

**TASK 1: Implement Tic Tac Toe game .**

import random

def print\_board(board):

for row in board:

print(" | ".join(row))

print("-" \* 9)

def empty\_spots(board):

return [(r, c) for r in range(3) for c in range(3) if board[r][c] == " "]

def check\_win(board, player):

win\_conditions = [

[(0, 0), (0, 1), (0, 2)], [(1, 0), (1, 1), (1, 2)], [(2, 0), (2, 1), (2, 2)],

[(0, 0), (1, 0), (2, 0)], [(0, 1), (1, 1), (2, 1)], [(0, 2), (1, 2), (2, 2)],

[(0, 0), (1, 1), (2, 2)], [(0, 2), (1, 1), (2, 0)]

]

for condition in win\_conditions:

if all(board[r][c] == player for r, c in condition):

return True

return False

def is\_draw(board):

return all(board[r][c] != " " for r in range(3) for c in range(3))

def evaluate(board):

if check\_win(board, "X"):

return 10 elif check\_win(board, "O"):

return -10

else:

return 0 def minimax(board, player, opponent, depth, is\_maximizing):

if check\_win(board, player):

return 10 - depth

elif check\_win(board, opponent):

return depth - 10

elif is\_draw(board):

return 0

if is\_maximizing:

best\_score = -float("inf")

for r, c in empty\_spots(board):

board[r][c] = player

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

board[r][c] = " "

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

return best\_score

else:

best\_score = float("inf")

for r, c in empty\_spots(board):

board[r][c] = opponent

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

board[r][c] = " "

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

return best\_score

def best\_move(board, player, opponent):

best\_score = -float("inf")

best\_move = None

for r, c in empty\_spots(board):

board[r][c] = player

score = minimax(board, opponent, player, 0, True)

board[r][c] = " "

if score > best\_score:

best\_score = score

best\_move = (r, c)

return best\_move

def tic\_tac\_toe():

board = [[" " for \_ in range(3)] for \_ in range(3)]

current\_player = "X"

opponent = "O"

while True:

print\_board(board)

if check\_win(board, "X"):

print("X wins!")

break

elif check\_win(board, "O"):

print("O wins!")

break

elif is\_draw(board):

print("It's a tie!")

break;

if current\_player == "X":

suggested\_move = best\_move(board, "X", "O")

print(f"Suggested move for 'X': {suggested\_move}")

row, col = map(int, input("Enter row and column numbers to place X (0-2, space separated): ").split())

if (row, col) in empty\_spots(board):

board[row][col] = "X"

else:

print("Invalid move. Try again.")

continue

else:

move = best\_move(board, "O", "X")

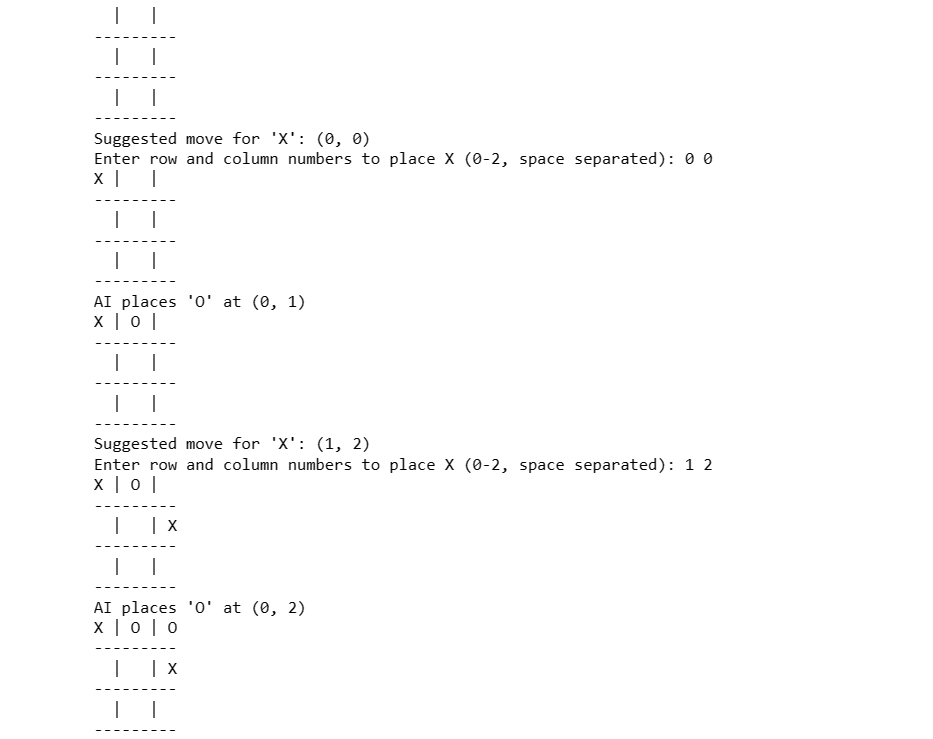
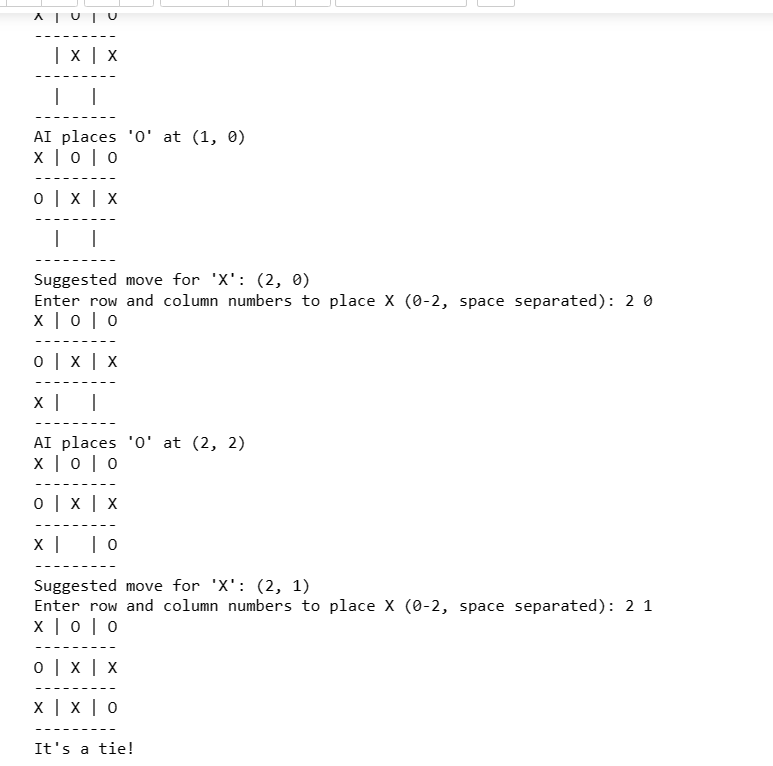
board[move[0]][move[1]] = "O"

print(f"AI places 'O' at {move}")

current\_player, opponent = opponent, current\_player

tic\_tac\_toe()

**OUTPUT:**

**** ****