Lab 1

Implement Tic –Tac –Toe Game.

Algorithm:

A sheet of paper with writing on it

AI-generated content may be incorrect.

Code:

def print\_board(board):

for row in board:

print(" ".join(row))

print()

def check\_winner(board, player):

for i in range(3):

if all(board[i][j] == player for j in range(3)):

return True

if all(board[j][i] == player for j in range(3)):

return True

if all(board[i][i] == player for i in range(3)):

return True

if all(board[i][2 - i] == player for i in range(3)):

return True

return False

def is\_draw(board):

return all(board[i][j] != '-' for i in range(3) for j in range(3))

cost\_counter = 0

def minimax(board, is\_ai\_turn):

global cost\_counter

cost\_counter += 1

if check\_winner(board, 'O'):

return 1

if check\_winner(board, 'X'):

return -1

if is\_draw(board):

return 0

if is\_ai\_turn:

best\_score = -float('inf')

for i in range(3):

for j in range(3):

if board[i][j] == '-':

board[i][j] = 'O'

score = minimax(board, False)

board[i][j] = '-'

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

return best\_score

else:

best\_score = float('inf')

for i in range(3):

for j in range(3):

if board[i][j] == '-':

board[i][j] = 'X'

score = minimax(board, True)

board[i][j] = '-'

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

return best\_score

def manual\_game():

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

print("Initial Board:")

print\_board(board)

while True:

while True:

try:

x\_row = int(input("Enter X row (1-3): ")) - 1

x\_col = int(input("Enter X col (1-3): ")) - 1

if board[x\_row][x\_col] == '-':

board[x\_row][x\_col] = 'X'

break

else:

print("Cell occupied!")

except:

print("Invalid input!")

print("Board after X move:")

print\_board(board)

if check\_winner(board, 'X'):

print("X wins!")

break

if is\_draw(board):

print("Draw!")

break

while True:

try:

o\_row = int(input("Enter O row (1-3): ")) - 1

o\_col = int(input("Enter O col (1-3): ")) - 1

if board[o\_row][o\_col] == '-':

board[o\_row][o\_col] = 'O'

break

else:

print("Cell occupied!")

except:

print("Invalid input!")

print("Board after O move:")

print\_board(board)

if check\_winner(board, 'O'):

print("O wins!")

break

if is\_draw(board):

print("Draw!")

break

global cost\_counter

cost\_counter = 0

cost = minimax(board, True)

print(f"AI evaluation cost from this position: {cost\_counter} states examined")

print(f"AI evaluation score from this position: {cost}")

manual\_game()

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

Implement vacuum cleaner agent.

Algorithm: