**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“JnanaSangama”, Belgaum -590014, Karnataka.**

****

**LAB REPORT**

**on**

**Artificial Intelligence (23CS5PCAIN)**

***Submitted by***

**Sareddy Poojya Sree (1BM23CS303)**

***in partial fulfillment for the award of the degree of***

**BACHELOR OF ENGINEERING**

***in***

**COMPUTER SCIENCE AND ENGINEERING**

****

**B.M.S. COLLEGE OF ENGINEERING**

**(Autonomous Institution under VTU)**

**BENGALURU-560019**

**Aug 2025 to Dec 2025**

**B.M.S. College of Engineering,**

**Bull Temple Road, Bangalore 560019**

(Affiliated To Visvesvaraya Technological University, Belgaum)

**Department of Computer Science and Engineering**

****

**CERTIFICATE**

This is to certify that the Lab work entitled “Artificial Intelligence (23CS5PCAIN)” carried out by **Sareddy Poojya Sree (1BM23CS303),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of an Artificial Intelligence (23CS5PCAIN) work prescribed for the said degree.

| Lab faculty Incharge Name  Assistant Professor  Department of CSE, BMSCE | Dr. Kavitha Sooda  Professor & HOD  Department of CSE, BMSCE |
| --- | --- |

**Index**

| **Sl.**  **No.** | **Date** | **Experiment Title** | **Page No.** |
| --- | --- | --- | --- |
| 1 | 30-9-2024 | Implement Tic –Tac –Toe Game  Implement vacuum cleaner agent |  |
| 2 | 7-10-2024 | Implement 8 puzzle problems using Depth First Search (DFS)  Implement Iterative deepening search algorithm |  |
| 3 | 14-10-2024 | Implement A\* search algorithm |  |
| 4 | 21-10-2024 | Implement Hill Climbing search algorithm to solve N-Queens problem |  |
| 5 | 28-10-2024 | Simulated Annealing to Solve 8-Queens problem |  |
| 6 | 11-11-2024 | Create a knowledge base using propositional logic and show that the given query entails the knowledge base or not. |  |
| 7 | 2-12-2024 | Implement unification in first order logic |  |
| 8 | 2-12-2024 | Create a knowledge base consisting of first order logic statements and prove the given query using forward reasoning. |  |
| 9 | 16-12-2024 | Create a knowledge base consisting of first order logic statements and prove the given query using Resolution |  |
| 10 | 16-12-2024 | Implement Alpha-Beta Pruning. |  |

Github Link:

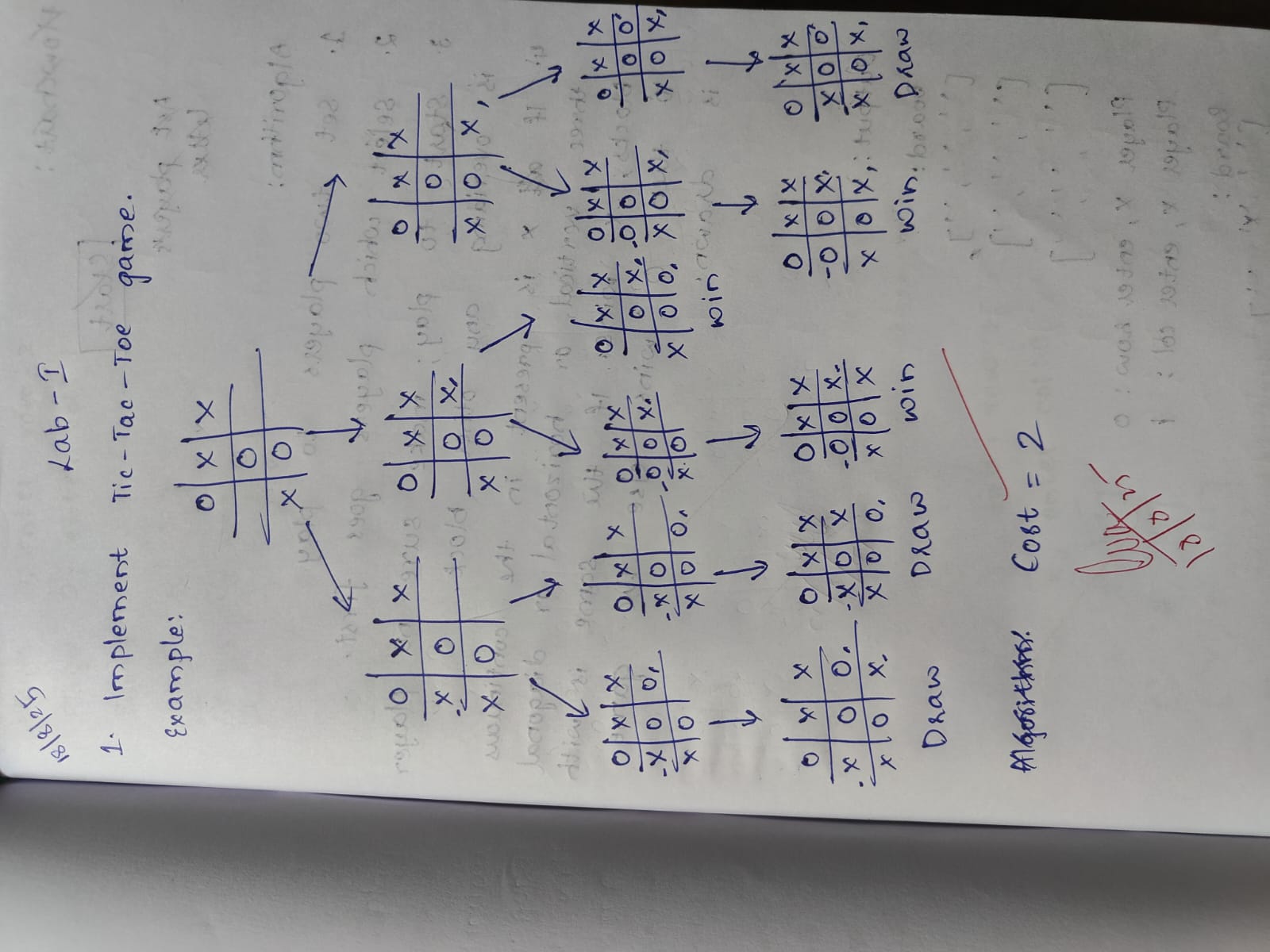
(You should provide your github link which contains all ten AI lab programs)

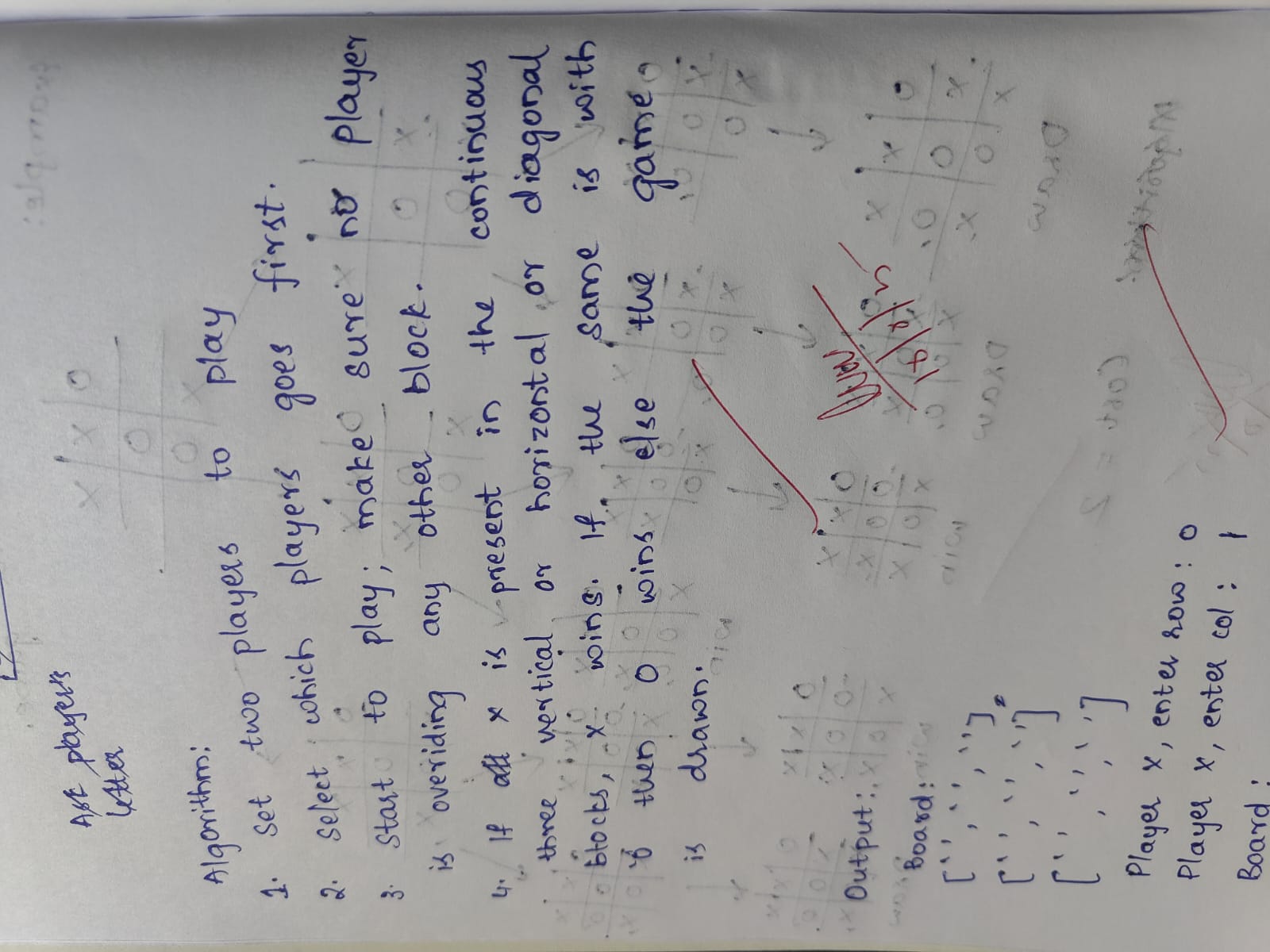
**Program 1**

Implement Tic –Tac –Toe Game

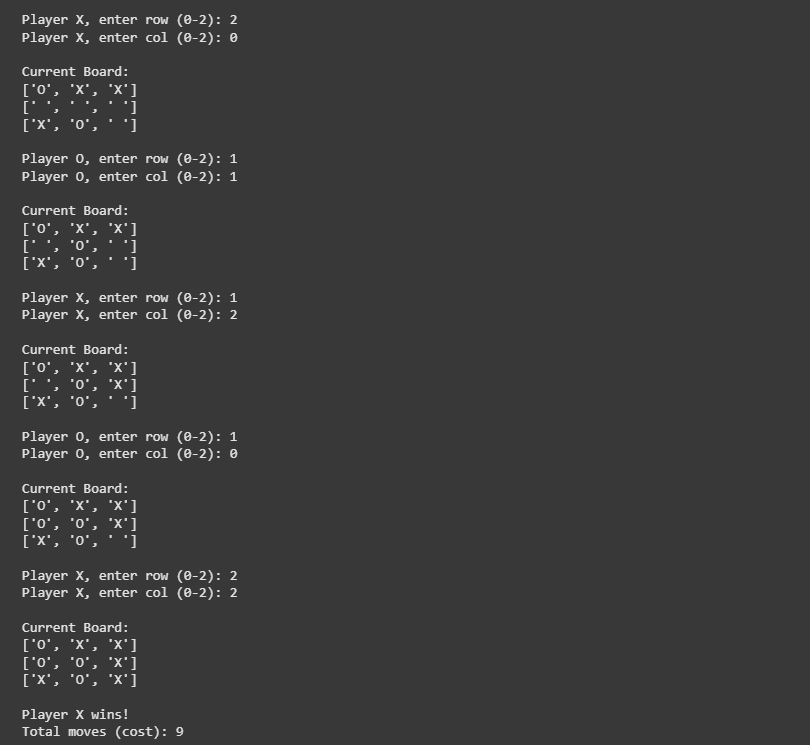
Implement vacuum cleaner agent

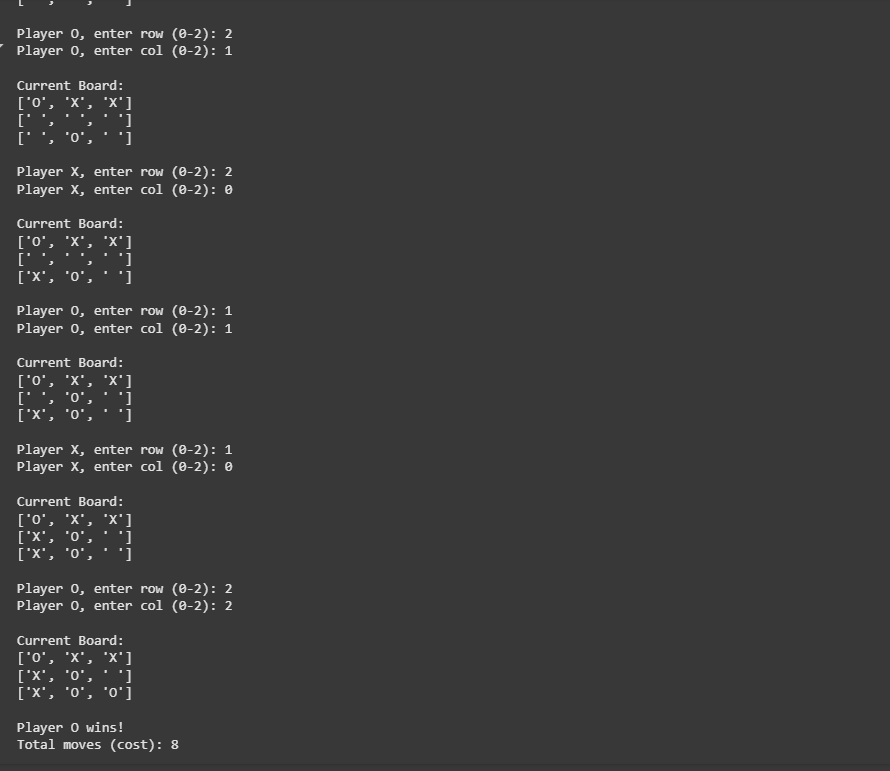
Algorithm:

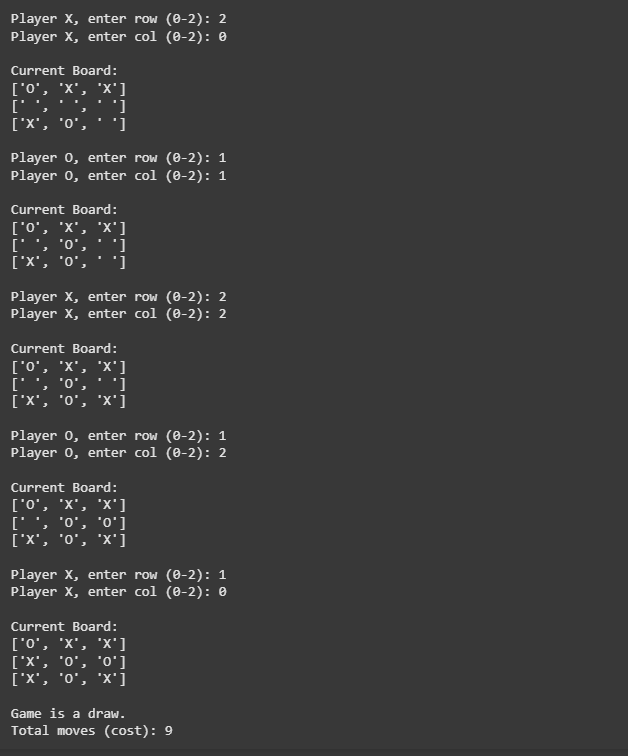




Output:







Code:

def print\_board(board):

print("\nCurrent Board:")

for row in board:

print(row)

print()

def check\_winner(board, player):

# Check rows

for row in board:

if all(cell == player for cell in row):

return True

# Check columns

for col in range(3):

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

return True

# Check diagonals

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

return True

return False

def tic\_tac\_toe():

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

players = ['X', 'O']

move\_count = 0

while True:

current\_player = players[move\_count % 2]

print(f"Player {current\_player}, enter row (0-2): ", end="")

row = int(input())

print(f"Player {current\_player}, enter col (0-2): ", end="")

col = int(input())

# If cell is empty

if board[row][col] == ' ':

board[row][col] = current\_player

move\_count += 1

print\_board(board)

if check\_winner(board, current\_player):

print(f"Player {current\_player} wins!")

print(f"Total moves (cost): {move\_count}")

break

if move\_count == 9: # Board full

print("It's a draw!")

break

else:

print("Cell already taken! Try again.")

# Run the game

tic\_tac\_toe()