**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

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

****

**LAB REPORT**

**on**

**Artificial Intelligence (23CS5PCAIN)**

***Submitted by***

**Suhas B P(1BM23CS345)**

***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 **Suhas B P (1BM23CS345),** 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: https://github.com/1BM23CS345/AI

**Program 1**

Implement Tic –Tac –Toe Game

Implement vacuum cleaner agent

Algorithm:

Implement Tic –Tac –Toe Game

Implement vacuum cleaner agent

Code:

Implement Tic –Tac –Toe Game

def print\_board(board):

for row in board:

print("|".join(row))

print()

def check\_winner(board, player):

for row in board:

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

return True

for col in range(3):

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

return True

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 is\_full(board):

return all(cell != " " for row in board for cell in row)

def play\_game():

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

current\_player = "X"

while True:

print\_board(board)

row, col = map(int, input(f"Player {current\_player}, enter row and col (0-2, space separated): ").split())

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

board[row][col] = current\_player

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

print\_board(board)

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

break

if is\_full(board):

print\_board(board)

print("It's a draw!")

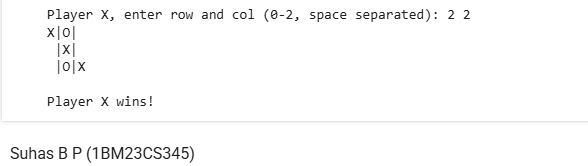
break

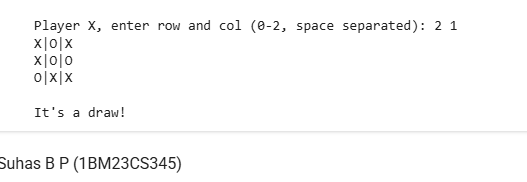
current\_player = "O" if current\_player == "X" else "X"

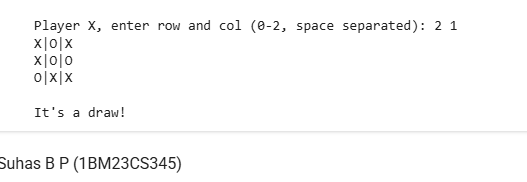
else:

print("Cell already taken, try again.")

play\_game()







Implement vacuum cleaner agent

rooms = {

'A': int(input("Enter state of A (0 for clean, 1 for dirty): ")),

'B': int(input("Enter state of B (0 for clean, 1 for dirty): ")),

'C': int(input("Enter state of C (0 for clean, 1 for dirty): ")),

'D': int(input("Enter state of D (0 for clean, 1 for dirty): "))

}

start = input("Enter starting location (A, B, C, or D): ").upper()

order = ['A', 'B', 'C', 'D']

if start not in order:

print("Invalid starting location.")

exit()

start\_index = order.index(start)

visited\_order = order[start\_index:] + order[:start\_index]

cost = 0

for i in range(len(visited\_order)):

current = visited\_order[i]

print(f"\nVacuum is in room {current}")

if rooms[current] == 1:

print(f"{current} is dirty.")

print(f"Cleaning {current}...")

rooms[current] = 0

cost += 1

else:

print(f"{current} is clean.")

if i < len(visited\_order) - 1:

next\_room = visited\_order[i + 1]

print(f"Moving vacuum to {next\_room}")

cost += 1

print(f"\nCost: {cost}")

print(rooms)

