**AI Lab Assignments to be submitted for Final Lab Exam**

**Programs to be executed in Python**

1. Breadth-First Search (BFS) Algorithm

Write a Python program to find the shortest path in an unweighted graph using BFS.

1. Depth-First Search (DFS) Algorithm

Implement DFS in Python to explore a graph and find paths or cycles.

1. A Search Algorithm\*

Develop a Python program to solve the shortest path problem using the A\* algorithm with heuristics.

1. Tic-Tac-Toe Game (AI vs Player using Minimax)

Build a Tic-Tac-Toe game where the AI uses the Minimax algorithm to play against a human player.

1. Sudoku Solver Using Backtracking

Write a Python program that solves a Sudoku puzzle using a backtracking algorithm.

1. 8-Puzzle Solver Using A Algorithm\*

Solve the 8-puzzle problem using A\* algorithm with a heuristic function like Manhattan distance.

1. Hill Climbing Algorithm

Implement the Hill Climbing algorithm to solve an optimization problem, such as maximizing a function.

1. Genetic Algorithm for Optimization

Write a Python program to solve an optimization problem using Genetic Algorithms.

1. N-Queens Problem

Solve the N-Queens problem using a backtracking algorithm in Python.

1. Bayesian Inference for Probabilistic Reasoning

Implement Bayesian inference in Python to predict the probability of a condition, like disease diagnosis.

1. Simple Chatbot Using Rule-Based Logic

Create a rule-based chatbot using Python that can respond to simple queries.

1. Decision Tree Classifier (Using Scikit-learn)

**Programs to be executed in Prolog**

1. Family Relationship Expert System

Write a Prolog program to represent a family tree and allow queries about relationships (e.g., parent, sibling, cousin).

1. Expert System for Medical Diagnosis

Develop a simple expert system in Prolog that diagnoses a disease based on symptoms provided by the user.

1. Solve the Monkey and Banana Problem

Implement the monkey and banana problem where the monkey needs to figure out how to get the banana using Prolog's logical reasoning.

1. 8-Puzzle Problem

Create a Prolog program that solves the 8-puzzle problem using a search strategy like depth-first or breadth-first search.

1. Towers of Hanoi Problem

Write a Prolog program to solve the Towers of Hanoi puzzle with recursive logic.

1. Traveling Salesman Problem (TSP)

Implement the Traveling Salesman Problem in Prolog using a brute-force search to find the shortest path.

1. Missionaries and Cannibals Problem

Solve the missionaries and cannibals problem in Prolog by ensuring that missionaries are never outnumbered by cannibals.

1. N-Queens Problem

Create a Prolog program to place N queens on an N×N chessboard such that no two queens threaten each other.

1. Water Jug Problem

Write a Prolog program to solve the water jug problem where two jugs with different capacities must measure a specific amount of water.

1. Prolog Chatbot

Develop a simple Prolog-based chatbot that can respond to user queries with predefined rule-based responses.

1. Arithmetic Puzzle Solver

Create a Prolog program that can solve puzzles like Sudoku, magic squares, or other arithmetic puzzles using constraint logic programming (CLP).

1. Logic Circuit Design

Write a Prolog program to simulate a basic logic circuit (e.g., AND, OR, NOT gates) and verify its truth table.

1. Route Finding System

Create a Prolog system to find routes between cities (nodes) using depth-first or breadth-first search algorithms.

1. Natural Language Processing (NLP)

Implement a simple natural language processor in Prolog that can parse and understand basic English sentences, focusing on grammar and syntax.