# Practical No :- 01

Aim :- To develop a prolog problem that represents a family tree and derived relationships like parent, child, sibling, grandparent etc.

Objectives :-
- To learn knowledge representation in the form of facts and rule.
- To understand how prolog uses logical inference to derive new relationships.
- To explore backtracking and pattern matching in prolog

Theory :-
A family tree represents relationships between members of a family, prolog is ideal for representing such relationship because it was

1) Facts :-
- facts describe direct relationship
father (ram, shyam)
mother ( sita, shyam)
father (ram, ~~guri)~~ gauri)
mother (sita, guari)

2) Rules :-
- Rules help derive indirect relationship
Eg. parent (x, y) - father (x, y)
parent (x, y) - mother (x, y)

grandparent (x,x) - parent (x,y), parent (y,z)

Working :-

- prolog uses backward chaining
- It starts from the query → matches with applicable rules → matches facts → deduces the answer

- This experiement helps understand symbolic AI and logical reasoning.

Conclusion :-

A family teee was successfully implemented using peolog facts and eules, demonsteating how logical eelationship can be deeived using infeeence.

Chhatrapati Shahu Maharaj Shikshan Sanstha's
CHH. SHAHU COLLEGE OF ENGINEERING
Kanchanwadi, Paithan Road, Chhatrapati Sambhajinagar Date :

# Practical No:-02

Aim :- To perform different operations on lists such as length, append, member, reverse using recursion in prolog.

Objective :-
- To learn recursion and pattern matching in prolog
- To perform operations like length, append, reverse and member checking

Theory :-
- prolog lists are recursive structure of the form [Head | Tail]
- List operations are performing using recursive rules

Eg.
- member (x, [x]-1)
- member (x, [-|T])) : member (x, T)

- This experiement improves understanding of declarative problem solving.

- List are the most powerful data structure in prolog
They are used for.
- search algorithm
- State representation
- Data manipulation
- Recursive processing

- Why lists are important
  - They represent sequences
  - used to store states (for AI search)
  - Handled naturally with recursive

key operations :-

1) Member checking
2) Appending
3) Reversing

Conclusion:-

prolog's recursive pattern matching effectively perform list operations and depends understanding of logic programming.

# Practical No :- 03

Aim :- To solve the 4L-3L water jug problem using breadth first search (BFS) in prolog

Objectives :-
- To understand state space representation
- To apply BFS for shortest-path discovery

Theory :- problem description
you have :-
- jug A = 4 liters
- jug B = 3 liters

Goal :- Measure 2 liter in any jug

1) State Representation :-
State is represented as :-
(A, B) = current amount in jug A and B

possible Moves :-
> fill a jug
> Empty a jug
> Transfer water from one jug to another

Example Moves :-
from (0,0) - fill A → (4,0)
from (4,0) - pour A to B → (1,3)
from (1,3) - Empty B → (1,0)

Why BFS :-

BFS checks states level by level, so it gives minimum steps solution

Eg. BFS Result path.
(0,0) → (4, 0) → (1,3) → (1,0) → (0, 1) → (4,1) → (2,3)
→ 2 liters achieved.

Conclusion :-
The BFS solution demostrate how prolog handles systematic exploration using queue based search to find optimal solution.

## Practical No :- 06

Aim :- To solve the same water jug problem using DFS search

Objectives :-

- To understand depth-first exploration
- To compare DFS with BFS

Theory :-

DFS characteristics :-

- Goes deep in one path
- uses recursion naturally
- may not give shortest path
- uses less memory
- Supported by prolog's - backtracking

Eg. DFS path.

(0,0) → (4,0) → (4,3), → (1,3) → (1,0) → ...... repeats deeper

DFS keeps going until goal is reached or deaded end occures.

DFS demonstrates how deep search works in AI and highlights the limitations of non-optimal search strategies

Comparison with BFS :-

| | BFS | DFS |
|---|---|---|
| 1) | finds shortest path | Not guaranteed |
| 2) | uses queue | uses stack / recursion |
| 3) | More memory | less Memory |
| 4) | complete | Incomplete |

Conclusion:-

DFS demonstrates how deep search works in AI and highlight the limitations of non-optimal search strategies.

## Practical No :- 5

Aim :- Write a prolog program to solve the tower of Hanoi puzzle.

Objectives :- To implement the towers of Hanoi problem in prolog using recursion and logical reasoning

Theory :-
The tower of Hanoi is a classic problem in computer science and AI that demonstrates the concept of recursion and problem-solving using logic.

It consist of three pages (A, B, C) and N disks of different size placed on Peg A in a decreasing order (largest at bottom)
The objective is to move all disks from peg A to peg C using peg B as an auxilliary, following these rules

- only one disk can be moved at a time
- A larger disk can't be placed on a smaller disk
- In prolog, recursion is used to express the solution logically.

- Move N-1 disk from source to auxiliary peg.
- Move the largest disk from auxiliary peg to destination peg

Example recursive rule structure

```
move (1, A →, C) :-
write('move disk from'), write(A), write('to'), write(C)
, nl,

move (N, A, B, C) :-
    N1 is N-1
    Move (N1, A, C, B)
    move (1, A, B, C)
    Move (N1, B, A, C)
```

This logic displays the sequence of moves required to solve the puzzles

Conclusion :-

This practical demonstrates how prolog can solve recursive problem like the towers of Hanoi using logical inference and recursion, showcasing prolog's strength in symbolic and AI - based problem - solving

Chhatrapati Shahu Maharaj Shikshan Sanstha's
CHH. SHAHU COLLEGE OF ENGINEERING
Kanchanwadi, Paithan Road, Chhatrapati Sambhajinagar Date :

# Practical No :- 06

Aim :- To write a prolog problem to place 8 queens on a dash (chess boards) without conflict.

Objectives :-
> understand constraint satisfaction problem (CSP)
> Apply backtracking to generate solution
> prevent quenss from attacking each other
> implement diagonal and column safety checks.

Theory :-
The 8 - queen problem is solved by placing one queen per row and checking.
- No two queens share same column.
- No two queens share same diagonal.

prolog tries permutations of 1-8 (column positions) and check constraints

AI Relevance
- CSP
- Backtracking
- search pruning.

- The 8 - queen problem is a classic constraint satisfaction and backtracking problem in AI

- The goal is to place 8 quenss on a standard 8x8 chessboard so that no two queens share the same row column and or diagonal.

- In prolog, this problem is solved using permutations and constraint checking
Each queen's position is represented by its column number in each row of the chessboard.
- Thus a solution can be represented as a list [Q1, Q2, Q3, Q4, Q5, Q6, Q7 Q8].
Where Qi denotes the column position of the queens in the i-th row.

Conclusion :-

This practical, demonstrates how prolog uses logical reasoning and backtracking to solve constraint satisfaction problem like the 8-queen puzzle efficiently exploring all valid configuration through recursion.

# Practical No :07

Aim :- To solve travelling salesman problem (TSP) using prolog by generating all paths and selecting the shortest.

Objectives :-

- understand optimization problem
- Represent cities and distance
- calculate route cost
- use permutation to generate solutions

Theory :-

- The travelling salesman problem (TSP) is a classic optimization and search problem in AI Artificial Intelligence.
- It Involves finding the shortest possible route that visits a sets of cities exactly once and return to the starting city

In prolog, this problem is represented using facts to define distance between cities and rules to calculate the total cost of each possible route
The prolog, this problem is represented using facts to define distance between cities and rules to calculate the total cost of each possible route

The language built in permutation and backtracking capabilities make it suitable for exploring all possible path systematically.

Approach:-
Represent cities and distance using facts such as :-

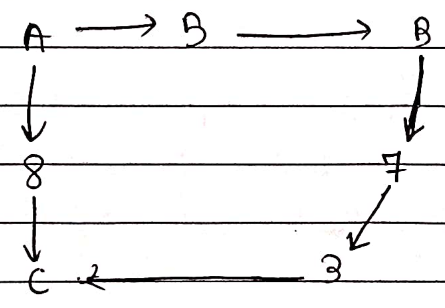
distance (a, b, 10)
distance (a, c, 15)
distance (b, c, 12)

- Use permutation to generate all possible tours calculate the total distance for each tour.

This approach ensures all paths are checked and the shortest one is selected.

Steps :-

1) Define distance as facts.
2) Generate permutation
3) Calculate total cost
4) choose minimum.



Conclusion :-

This practical introduced combinational optimization and brute - force search using prolog

# Practical No :- 08

Aim :- To solve the 8- puzzle game using state -space search in prolog

Objective :-
- understand sliding the puzzle (tile)
- Implement state transition
- Apply BFS / DFS search
- Learn heuristic - based problem solving

Theory :-
The puzzle consist of tiles numbered 1-8 and one blank

| | | |
|---|---|---|
| 1, | 2, | 3, |
| 4, | 0, | 5, |
| 6, | 7, | 8, |

- Move blank up
- move Down
- move left
- move Right

Search :-
> BFS → shortest solution
> DFS → Deep Search
> A* → heuristic search

- Heuristic :-

Manhatten distance = sum of tile distances from goal

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | _ | 5 |
| 6 | 7 | 8 |

This demonstrates how prolog's recursive search and logical reasoning can systematically explore possible moves until the goal is achieved.

Conclusion :-

8-puzzle demonstrate state-space search heuristics and problem solving technique used in AI for real-world planning and decision making.