| **Name : Soham.M.Shetty** | **Class/Roll No. : D11AD/58** | **Grade : 5th Sem** |
| --- | --- | --- |

**Title of Experiment :** Write simple programs using PROLOG as an AI programming Language**.**

**Objective of Experiment :** Understand and write programmes on Prologue for the given problem statements .

**Outcome of Experiment :** Understands the basic of Prolog and its implication , write prolog code on SWI- Prolog for the given problem statement.

**Problem Statement :** Write simple programs using PROLOG as an AI programming Language.

1) Create a Family Tree.

2)Solve the Tower of Hanoi Problem

**Description / Theory :**

SWI-Prolog is a widely used implementation of the Prolog programming language. Prolog stands for "Programming in Logic," and it is a declarative language used for symbolic reasoning and problem-solving based on formal logic. SWI-Prolog provides an interactive environment where you can define facts, rules, and queries. It allows you to work with knowledge bases and perform logical inference to solve problems. Prolog programs consist of facts, which state information about the domain, and rules, which define relationships and logical implications.

Here's a brief overview of some key concepts and features of Prolog:

* Facts and Rules: In Prolog, you define facts as statements about objects and their relationships. Rules are used to define logical implications or relationships between different facts. Facts and rules are the building blocks of Prolog programs.
* Variables: Variables in Prolog start with an uppercase letter or an underscore. They act as placeholders that can be matched with any value during the execution of queries.
* Queries: You can pose queries to the Prolog system to find solutions to specific problems. Prolog uses unification and backtracking to search for solutions that match the specified criteria.

Family tree:

In Prolog, **a family tree** can be represented using facts and rules to define the relationships between family members. Each family member is represented as an individual, and their relationships with others are captured through predicates and rules.

\* Facts: Parent-Child Relationships

We define facts to represent parent-child relationships. For example, parent(john, mary) states that "John is the parent of Mary." Similarly, parent(mary, anne) states that "Mary is the parent of Anne."

\*Rules: Inferring Other Relationships

We can use rules to infer more complex relationships from the given facts. For

Example:

1)father(Father, Child) :- parent(Father, Child), male(Father) states that "Father is the father of Child if Father is the parent of Child and Father is male."

2)mother(Mother, Child) :- parent(Mother, Child), female(Mother) states that "Mother is the mother of Child if Mother is the parent of Child and Mother is female."

3)grandparent(Grandparent, Grandchild) :- parent(Grandparent, Parent), parent(Parent, Grandchild) states that "Grandparent is the grandparent of Grandchild if there exists a Parent such that Grandparent is the parent of Parent and Parent is the parent of Grandchild."

\*Facts: Gender Information

To distinguish between male and female family members, we can define facts that state their genders. For example, male(john) states that "John is male," and female(mary) states that "Mary is female."

With these facts and rules, we can query the Prolog system to explore the family tree. For example, we can ask questions like "Who is Mary's father?" or "Who are the cousins of David?" Prolog will use unification and backtracking to find the answers based on the defined relationships.

The family tree example in Prolog provides a powerful way to represent and reason about complex relationships between family members, and it demonstrates the declarative nature of Prolog, where relationships are defined logically rather than procedurally.

Tower of Hanoi Problem:

**Towers of Hanoi Problem** is a famous puzzle to move N disks from the source peg/tower to the target peg/tower using the intermediate peg as an auxiliary holding peg. There are two conditions that are to be followed while solving this problem −

A larger disk cannot be placed on a smaller disk.

Only one disk can be moved at a time.

To solve this, we have to write one procedure move(N, Source, Target, auxiliary). Here N number of disks will have to be shifted from Source peg to Target peg keeping Auxiliary peg as intermediate.

For example – move(3, source, target, auxiliary).

Move top disk from source to target

Move top disk from source to auxiliary

Move top disk from target to auxiliary

Move top disk from source to target

Move top disk from auxiliary to source

Move top disk from auxiliary to target

Move top disk from source to target

The recursive nature of the program reflects the nature of the Tower of Hanoi problem, where a larger problem is broken down into smaller, identical subproblems until a base case is reached (in this case, moving a single disk). The program then prints the steps to move each disk from the source peg to the destination peg.

**Program:**

/\*Family tree\*/

/\*facts\*/

male(timanna).

male(manjappa).

male(mudanna).

male(sena).

male(ravindra).

male(muks).

male(soham).

male(shaurya).

female(suvarna).

female(jalaja).

female(shweta).

parent\_of(manjappa,sena).

parent\_of(timanna,ravindra).

parent\_of(mudanna,suvarna).

parent\_of(ravindra,muks).

parent\_of(suvarna,muks).

parent\_of(jalaja,shweta).

parent\_of(sena,shweta).

parent\_of(muks,soham).

parent\_of(muks,shaurya).

parent\_of(shweta,soham).

parent\_of(shweta,shaurya).

/\*rules\*/

mother(X,Y):- parent\_of(X,Y),female(X).

father(X,Y):- parent\_of(X,Y),male(X).

haschild(X):- parent\_of(X,\_).

brother(X,Y):- parent\_of(Z,X),parent\_of(Z,Y),male(X),X\==Y.

grandparent(X,Y):- parent\_of(X,Z),parent\_of(Z,Y).

grandmother(X,Z):- mother(X,Y),parent\_of(Y,Z).

grandfather(X,Z):-father(X,Y),parent\_of(Y,Z).

wife(X,Y):-parent\_of(X,Z),parent\_of(Y,Z),female(X),male(Y).

husband(X,Y):-parent\_of(X,Z),parent\_of(Y,Z),female(Y),male(X).

g\_grandparent(X,Y):- parent\_of(X,W),parent\_of(W,Z),parent\_of(Z,Y).

/\*Tower of Hanoi Problem \*/

move(1,X,Y,\_) :-

write('Move top disk from '), write(X), write(' to '), write(Y), nl.

move(N,X,Y,Z) :-

N>1,

M is N-1,

move(M,X,Z,Y),

move(1,X,Y,\_),

move(M,Z,Y,X).

| 1. | **Family tree : Output Screenshots :** |
| --- | --- |
| 2. | **Tower of Hanoi :**  **Output Screenshots :** |

**Results and Discussions :** Here we studied and implemented programs on Prolog , famous problem statements like family tree and tower of hanoi problem in prolog . We implemented it in SWI-Prolog , understood its working and how the program works . And we successfully implemented the program.