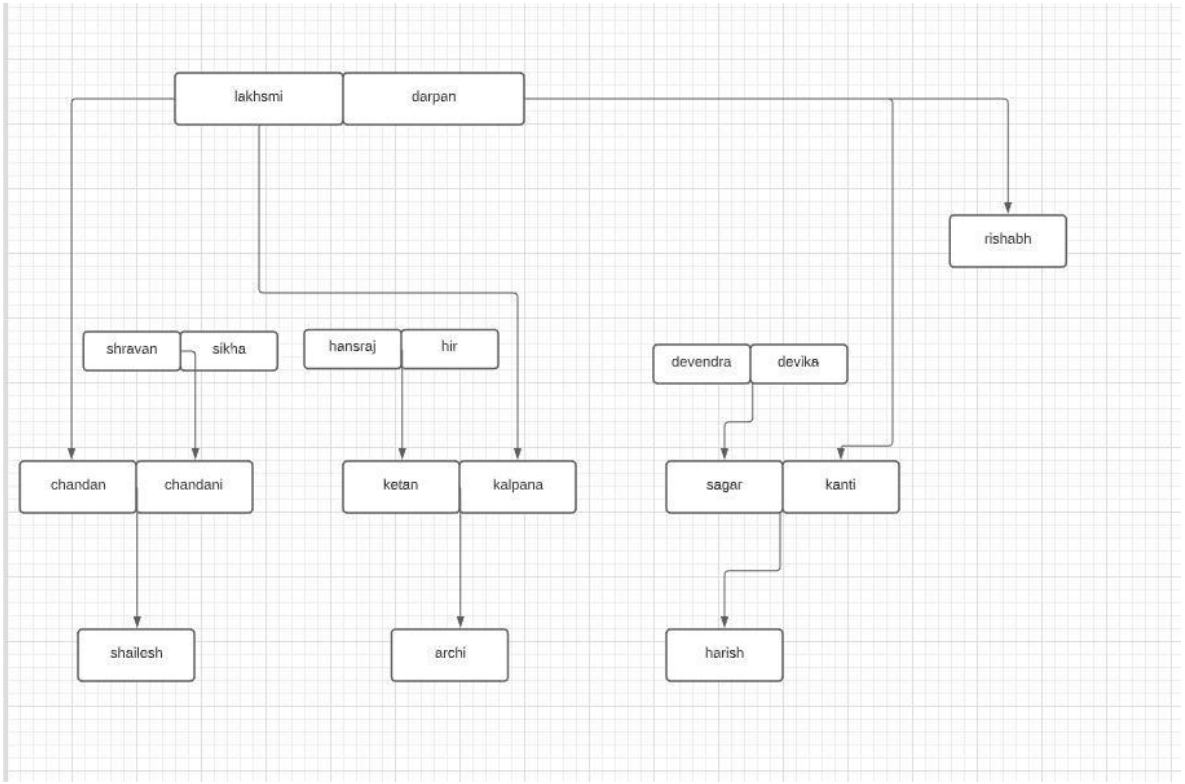
Experiment No. 5
Title: Family Tree in PROLOG using condition action rule based agent

Batch: A4 **Roll No.:** 1914072 **Experiment No.:** 5

Aim: Write a program for implementation of family tree in PROLOG using condition-action rules based agent.

Family Tree:



Results:

Code:

```
female(sikha).
female(hir).
female(devika).
female(archi).
female(kalpana).
female(chandani).
female(kanti).
female(lakhsmi).

male(sagar).
male(shailesh).
male(darpan).
male(harish).
male(chandan).
male(ketan).
male(rishabh).
male(shravan).
male(hansraj).
male(devendra).

parent_of(shravan,chandani).
parent_of(sikha,chandani).
parent_of(hansraj,ketan).
parent_of(hir,ketan).
parent_of(devendra,sagar).
parent_of(devika,sagar).
parent_of(sagar,harish).
parent_of(kanti,harish).
parent_of(kalpana,archi).
parent_of(ketan,archi).
parent_of(chandan,shailesh).
```

```
parent_of(chandani,shailesh).
parent_of(lakhsmi,kanti).

parent_of(darpan,kanti).
parent_of(lakhsmi,kalpana).
parent_of(darpan,kalpana).
parent_of(lakhsmi,chandan).
parent_of(darpan,chandan).
parent_of(lakhsmi,rishabh).
parent_of(darpan,rishabh).

married(shravan,sikha).
married(hansraj,seena).
married(devendra,devika).
married(sagar,kanti).
married(kanti,sagar).
married(chandan,chandani).
married(chandani,chandan).
married(ketan,kalpana).
married(kalpana,ketan).
married(lakhsmi,darpan).
married(darpan,lakhsmi).
child_of(X,Y):- parent_of(Y,X).
spouse_of(X,Y):- married(X,Y).
father_of(X,Y):- male(X),
parent_of(X,Y).
mother_of(X,Y):- female(X),
parent_of(X,Y).
maternal_grandfather_of(X,Y):- male(X),
mother_of(Z,Y), father_of(X,Z).
paternal_grandfather_of(X,Y):- male(X),
father_of(Z,Y), father_of(X,Z).
maternal_grandmother_of(X,Y):- female(X),
mother_of(Z,Y), mother_of(X,Z).
```

```
paternal_grandmother_of(X,Y):- female(X),
father_of(Z,Y), mother_of(X,Z).
sister_of(Y,X):-
female(X),
father_of(F, Y), father_of(F,X),X \= Y.
sister_of(X,Y):- female(X),
mother_of(M, Y), mother_of(M,X),X \= Y.
aunt_of(X,Y):-
mother_of(Z,Y), sister_of(Z,X).
aunt_of(X,Y):-
father_of(Z,Y), sister_of(Z,X).
brother_of(X,Y):-
male(X),
father_of(F, Y), father_of(F,X),X \= Y.
brother_of(X,Y):- male(X),
mother_of(M, Y), mother_of(M,X),X \= Y.
uncle_of(X,Y):-
father_of(Z,Y), brother_of(Z,X).
uncle_of(X,Y):-
mother_of(Z,Y), brother_of(X,Z).
cousin_of(X,Y):-
uncle_of(Z,X), father_of(Z,Y).
```

Output:

```
For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- sister_of(X,Y).
X = kanti,
Y = kalpana ;
X = chandan,
Y = kalpana ;
X = rishabh,
Y = kalpana ;
X = kalpana,
Y = kanti ;
X = chandan,
Y = kanti ;
X = rishabh,
Y = kanti ;
X = kalpana,
Y = kanti ;
X = kalpana,
Y = chandan ;
X = kalpana,
Y = rishabh ;
X = kanti,
Y = kalpana ;
X = kanti,
Y = chandan ;
X = kanti,
Y = rishabh ;
false.
```

```
?- brother_of(X,Y).
X = chandan,
Y = kanti ;
X = chandan,
Y = kalpana ;
X = chandan,
Y = rishabh ;
X = rishabh,
Y = kanti ;
X = rishabh,
Y = kalpana ;
X = rishabh,
Y = chandan ;
X = chandan,
Y = kanti ;
X = chandan,
Y = kalpana ;
X = chandan,
Y = rishabh ;
X = rishabh,
Y = kanti ;
X = rishabh,
Y = kalpana ;
X = rishabh,
Y = chandan ;
false.
```

```
?- uncle_of(X,Y).
X = kanti,
Y = shailesh ;
X = kalpana,
Y = shailesh ;
X = rishabh,
Y = shailesh ;
X = kanti,
Y = shailesh ;
X = kalpana,
Y = shailesh ;
X = rishabh,
Y = shailesh ;
X = chandan,
Y = archi ;
X = rishabh,
Y = archi ;
X = chandan,
Y = archi ;
X = rishabh,
Y = archi ;
X = chandan,
Y = harish ;
X = rishabh,
Y = harish ;
X = chandan,
Y = harish ;
X = rishabh,
Y = harish ;
false.
```

```
?- cousin_of(X,Y).
X = archi,
Y = shailesh ;
X = archi,
Y = shailesh ;
X = harish,
Y = shailesh ;
X = harish,
Y = shailesh ;
false.
```

```
?- paternal_grandfather_of(X,Y).
X = darpan,
Y = shailesh ;
X = hansraj,
Y = archi ;
X = devendra,
Y = harish ;
false.
```

```
?- paternal_grandmother_of(X,Y).
X = hir,
Y = archi ;
X = devika,
Y = harish ;
X = lakhsmi,
Y = shailesh ;
false.
```

Questions:

1. The PROLOG suit is based on

a. Interpreter

b. Compiler

c. None of the above

d. Both

Ans: (d) Both.

2. State true of false

There must be at least one fact pertaining to each predicate written in the PROLOG program.

Ans: True.

3. State true of false

In PROLOG program the variable declaration is a compulsory part.

Ans: False

4. Differentiate between a fact and a predicate with syntax.

Ans: Facts:

Facts are the statements which is used to describe properties of or between objects. A fact is an expression that makes a declarative statement about the problem domain. Facts have simple rules of syntax. They should always begin with lowercase letter and end with a full stop.

Syntax:

```
male(sam).
female(tina).
```

Here, the statements male(sam) and female(tina) are describing the property of objects herb and mona

respectively and hence are facts.

Predicate:
Both facts and rules are Predicate definitions. A predicate denotes a property or relationship between objects. It is the name given to the word occurring before the bracket of fact or rule.
Syntax:

```
spouse(sam,tina).
sibling(sam,tina).
```

Here, the words spouse and sibling are predicates.

5. Differentiate between knowledge base and Rule base approach.

Ans: Rule-based systems use a rule-based approach and process data and output information, but they also process rules and make decisions. They are good at processing lots of simple business rules with broad logic. They are commonly used for real-time decisioning systems and compliance systems.

Knowledge-based systems use a knowledge-based approach and also process data and rules to output information and make decisions. In addition, they also process expert knowledge to output answers, recommendations, and expert advice. They are good at processing deep logic and very complex business rules. They are commonly used for advising systems, expert systems, and knowledge automation.

	Rule-based	Knowledge-based
Can process	Data Rules	Data Rules Knowledge
Can output	Information Decision Real-time decision	Information Decision Answers Expert advice Recommendations
Commonly used for	Enterprise rules	Departmental rules
Ideal for	Simplistic business rules	Complex business rules
Best for these types of applications	Decisioning Compliance	Advising Product selection Recommending Troubleshooting
Domain scope	Broad logic	Deep logic

Outcomes: CO3: Family tree in PROLOG using condition-action rules-based agent.

Conclusion: We studied and understood about PROLOG programming and implemented a family tree using the same**.**

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of faculty in-charge with date

References:

- Stuart Russell and Peter Norvig, Artificial Intelligence: A Modern Approach, Second Edition, Pearson Publication
- Elaine Rich, Kevin Knight, Artificial Intelligence, Tata McGraw Hill, 1999.