Shri Vile Parle Kelavani Mandal'

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Name: Pagariya Sakshi **Roll No**. 17

Subject : Artificial Intelligence Lab **Subject Code :** BTCOL707

Class: Final Year Comp. Engg. **Expt. No. :** 01

Title : Study of PROLOG. Write the following programs using PROLOG.

Problem Staement :

Study of PROLOG. Write the following programs using PROLOG.

1) Write simple fact for following:

a. Ram likes mango.

b. Seema is a girl.

c. Bill likes Cindy.

d. Rose is red.

e. John owns gold

2) Write predicates One converts centigrade temperatures to Fahrenheit, the other checks if a temperature is below freezing.

Software Required :

Prolog

Theory :

PROLOG-PROGRAMMING IN LOGIC

PROLOG stands for Programming, In Logic — an idea that emerged in the early 1970's to use logic as programming language. The early developers of this idea included Robert Kowaiski at Edinburgh (on the theoretical side), Marrten van Emden at Edinburgh (experimental demonstration) and Alian Colmerauer at Marseilles (implementation). David D.H. Warren's efficient implementation at Edinburgh in the mid -1970's greatly contributed to the popularity of PROLOG. PROLOG is a programming language centered around a small set of basic mechanisms, including pattern matching, tree based data structuring and automatic backtracking. This Small set constitutes a surprisingly powerful and flexible programming framework. PROLOG is especially well suited for problems that involve objects- in particular, structured objects- and relations between them.

SYMBOLIC LANGUAGE

PROLOG is a programming language for symbolic, non-numeric computation. It is especially well suited for solving problems that involve objects and relations between objects. For example, it is an easy exercise in prolog to express spatial relationship between objects, such as the blue sphere is behind the green one. It is also easy to state a more general rule: if object X is closer to the observer than object Y. and object Y is closer than Z, then X must be closer than Z. PROLOG can reason about the spatial relationships and their consistency with respect to the general rule. Features like this make PROLOG a powerful language for Artificia1 Language (AL) and non- numerical programming.

There are well-known examples of symbolic computation whose implementation in other standard languages took tens of pages of indigestible code, when the same algorithms were implemented in PROLOG, the result was a crystal-clear program easily fitting on one page.

FACTS, RULES AND QUERIES

Programming in PROLOG is accomplished by creating a database of facts and rules about objects, their properties, and their relationships to other objects. Queries then can be posed about the objects and valid conclusions will be determined and returned by the program Responses to user queries are determined through a form of inference control known as resolution. To define and modify knowledge, you work with facts, rules, and queries in the logic programming language Prolog. Horn clauses are a type of symbolic logic used in Prolog. These core ideas are explained as follows:Details

Facts in Prolog are statements in the form of predicates that characterize some fundamental information or relationships. Things that are true are asserted using facts.

A predicate and a collection of arguments make up a fact. For instance:

```
male(john).
female(lisa).
parent(john, lisa).
```

RULES

Relationships and conclusions drawn from facts and other rules are expressed using rules. They are made up of a body and a head.

The conclusion or intended inference is the head of a rule.

The requirements or subgoals that must be met in order for a rule to be applied make up its body.

Here is an illustration of a rule:

```
father(X, Y) :- male(X), parent(X, Y).
```

In this rule, "X is the father of Y" if X is male, and X is a parent of Y.

The main method of communicating with a Prolog system is through queries. They let you pose queries or ask questions in accordance with the established guidelines and facts.

In order to obtain answers, Prolog will try to combine the goals that are provided for each query with the facts and rules.

As an illustration, you may question something like:

```
?- father(john, lisa).
```

Based on the established rules and data, Prolog will attempt to determine whether or not John is Lisa's father in response to this query.

This is an example that shows how to use facts, rules, and queries in Prolog in its entirety:

```
% Facts
male(john).
female(lisa).
parent(john, lisa).
% Rules
father(X, Y) :- male(X), parent(X, Y).
mother(X, Y) :- female(X), parent(X, Y).
% Query
?- father(john, lisa). % This will return true.
```

In order to determine if the question is true or false, Prolog will analyze it by searching for a set of facts and rules that match the query. It accomplishes this through a procedure known as "resolution". In the event that a solution is found, it will also yield variable values.

PROLOG IN DESIGNING EXPERT SYSTEMS

An expert system is a collection of programs designed to manipulate knowledge that has been encoded in order to solve issues in a specific field where human competence is typically needed. Knowledge for an expert system is gathered from authoritative sources like texts, journal articles, databases, etc. and encoded in a format that the system can utilize for inference or reasoning. After acquiring a sufficient body of expert knowledge, it must be encoded in some way, loaded into a knowledge base, tested, and continually improved during the system's lifespan. PROLOG is an effective language for creating expert systems due to the following characteristics.

- Use of knowledge rather than data
- Modification of the knowledge base without recompilation of the control programs.
- Capable of explaining conclusion.
- Symbolic computations resembling manipulations of natural language.
- Reason with meta-knowledge.

META PROGRAMMING

A program that uses other programs as data is called a meta-program. Meta-programs include, for example, interpreters and compilers. One type of meta-program is a meta-interpreter, which is an interpreter written in a language for another language. Hence, an interpreter for PROLOG that is written in PROLOG is called a PROLOG interpreter. PROLOG's ability to manipulate symbols makes it an effective language for meta-programming. As a result, it is frequently employed as a language for language implementation. When it comes to fast implementing new ideas, PROLOG is an especially good language for rapid prototyping. Innovative concepts are tried out and implemented quickly.

	1)Programe
	red(rose).
	likes(bill ,cindy).
	owns(john ,gold).
	Output:
	Goal
	queries
	?-likes(ram,What).
	What= mango
	?-likes(Who,cindy).
	Who= cindy
	?-red(What).
	What= rose
	?-owns(Who,What).
	Who= john
	What= gold.
	2) Program:
	Production rules:
	Arithmetic:
	c_to_f f is c * 9 / 5 +32
	freezing f < = 32
	Rules:
	c_to_f(C,F) :-
	F is C * 9 / 5 + 32.
	freezing(F) :-
Conclusion:	F =< 32.

Output:

Queries:

```
?- c_to_f(100,X).

X = 212

Yes

?- freezing(15)

.Yes

?- freezing(45).

No
```