Name:	Sunit Sunil Khaire
Roll No:	19
Class/Sem:	TE/V
Experiment No.:	7
Title:	Implement knowledge base in Prolog.
Date of Performance:	
Date of Submission:	
Marks:	
Sign of Faculty:	

Aim: Implement knowledge base in Prolog.

Objective: To study and use AI programming language to create knowledge base.

Theory:

Prolog is a logic programming language. It has important role in artificial intelligence. Unlike many other programming languages, Prolog is intended primarily as a declarative programming language. In prolog, logic is expressed as relations (called as Facts and Rules). Core heart of prolog lies at the logic being applied. Formulation or Computation is carried out by running a query over these relations.

In prolog, We declare some facts. These facts constitute the Knowledge Base of the system. We can query against the Knowledge Base. We get output as affirmative if our query is already in the knowledge Base or it is implied by Knowledge Base, otherwise we get output as negative. So, Knowledge Base can be considered similar to database, against which we can query. Prolog facts are expressed in definite pattern. Facts contain entities and their relation. Entities are written within the parenthesis separated by comma (,). Their relation is expressed at the start and outside the parenthesis. Every fact/rule ends with a dot (.)

Take any problem and represent the knowledge (facts) in prolog. Also you can use this for reasoning purpose.

SWI-Prolog offers a comprehensive free Prolog environment. Since its start in 1987, SWI-Prolog development has been driven by the needs of real world applications. SWI-Prolog is widely used in research and education as well as commercial applications.

SWI-Prolog, **a free implementation of the programming language Prolog**. Susceptibility weighted imaging, in magnetic resonance imaging (MRI) used in medical contexts.

Logic píogíamming languages, of which PROLOG (programming in logic) is the best known, state a program as a set of logical relations (e.g., a grandparent is the parent of a parent of someone). Such languages are similar to the SQL database language. A program is executed by an "inference engine" that answers a query by searching these relations systematically to make infeíences that will answer a query. PROLOG has been used extensively in natural language processing and other AI programs.

Example: The problem of murder mystery.

Five persons Alice, her husband, brother, son and daughter

Event: One murder. One of the five is victim and one is Killer.

Rules:

1) Husband and Alice was not together on the night of murder.

2) The killer and victim were on the beach.

3) On the night of murder, one male and one female was in the bar.

4) The victim was twin and the counterpart was innocent.

5) The killer was younger than the victim.

6) One child was alone at home.

Code for Prolog problem of murder mystery in Artificial Intelligence:

predicates

```
%   pair(symbol,symbol)
    iskiller(symbol,symbol)
    male(symbol)
    female(symbol)
    isvictim(symbol)
    not_at_bar(symbol,symbol)
    not_at_beach(symbol,symbo
    l) not_alone(symbol)
    twin(symbol,symbol)
    younger(symbol,symbol)
    child(symbol)
```

clauses

```
    male(husband)
    .
    male(brother).
    male(son).
    female(alice).
    female(daughter).
    twin(brother,alice).
    twin(son,daughter).

    child(son).
    child(daughteí).
```

Program

```
    /* Facts */

    male(jack).

    male(oliver).

    male(ali).

    male(james).
```

```
male(simon).
male(harry).
female(helen).
female(sophie).
female(jess).
female(lily).

parent_of(jack,jess).
parent_of(jack,lily).
parent_of(helen, jess).
parent_of(helen, lily).
parent_of(oliver,james).
parent_of(sophie, james).
parent_of(jess, simon).
parent_of(ali, simon).
parent_of(lily, harry).
parent_of(james, harry).

/* Rules */
father_of(X,Y):- male(X),
   parent_of(X,Y).

mother_of(X,Y):- female(X),
   parent_of(X,Y).

grandfather_of(X,Y):- male(X),
   parent_of(X,Z),
   parent_of(Z,Y).

grandmother_of(X,Y):- female(X),
```

```
    parent_of(X,Z),
    parent_of(Z,Y).

sister_of(X,Y):- %(X,Y or 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):- female(X),
    parent_of(Z,Y), sister_of(Z,X),!.

brother_of(X,Y):- %(X,Y or Y,X)%
    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):-
    parent_of(Z,Y), brother_of(Z,X).

ancestor_of(X,Y):- parent_of(X,Y).
ancestor_of(X,Y):- parent_of(X,Z),
    ancestor_of(Z,Y).
```

Output:

```
parent_of(X,jess).
```

Conclusion:

The Prolog experiment effectively showcased the language's strengths in representing and querying relational data through facts and rules. By constructing a family tree, we demonstrated Prolog's ability to reason about relationships, such as parentage and sibling connections. This highlighted Prolog's efficiency in handling complex logical structures, making it a valuable tool for applications in artificial intelligence and expert systems. Overall, the experiment underscored Prolog's unique approach to programming through logical inference and symbolic reasoning.