St. Francis Institute of Technology
Borivli (West), Mumbai-400103
Department of Information Technology

Experiment – 10

1. Aim: To implement knowledge base, facts and rules

2. Objective: After performing the experiment, the students will be able to implement

- Define facts and rules
- Create knowledge base

3. Lab objective mapped: To understand, formulate and implement declarative programming paradigm through logic programming (PSO2) (PO1)

4. Prerequisite: Nil

5. Requirements: The following are the requirements – Prolog Compiler

6. Pre-Experiment Theory:

Facts are statements about what is true about a problem, instead of instructions how to accomplish the solution.
The Prolog system uses the facts to work out how to accomplish the solution by searching through the space of possible solutions.
It is defined by an identifier followed by an n-tuple of constants.
A relation identifier is referred to as a predicate
When a tuple of values is in a relation we say the tuple satisfies the predicate.
Rules specify under what conditions a tuple of values satisfies a predicate.
The basic building block of a rule is called an atom

7. Laboratory Exercise

A. Steps to be implemented

1. Open SWI-Prolog
2. Go to ....file-> new
3. New prolog editor will open up.
4. Write your program (collection of facts and rules)
5. Save the file in the desired as .pl file
6. Follow the steps -> Save buffer, Compile buffer and Make
7. At the prompt, first change to working directory using cd command
8. Load the required file
9. To check the outputs, fire appropriate queries at the prompt

B. Program Code

1. WAP in prolog to create knowledge base using the following facts

woman(mia).
woman(jody).
woman(yolanda).
playsAirGuitar(jody).
For the above knowledge base, fire the following queries-woman(mia)., woman(jody). woman(yolanda). playsAirGuitar(jody). playsAirGuitar(mia).

```
1 ?- cd('D:/College/PCPF/Prolog/Experiments/Exp10').
true.

2 ?- consult('Exp10_1.pl').
true.

3 ?- woman(mia).
true.

4 ?- woman(jody).
true.

5 ?- woman(yolanda).
true.

6 ?-  playsAirGuitar(jody).
true.

7 ?- playsAirGuitar(mia).
false.

8 ?-
```

2. 1. WAP in prolog to create knowledge base using the following facts and rules

happy(yolanda).
listens2Music(mia).
listens2Music(yolanda):- happy(yolanda).
playsAirGuitar(mia):- listens2Music(mia).
playsAirGuitar(yolanda):- listens2Music(yolanda).
For the above knowledge base, fire the following queries- happy(yolanda)., happy(mia)., listens2Music(mia)., listens2Music(yolanda)., happy(X), playsAirGuitar(Y)

```
1 ?- cd('D:/College/PCPF/Prolog/Experiments/Exp10').
true.

2 ?- consult('Exp10_2.pl').
true.

3 ?- happy(yolanda).
true.

4 ?- happy(mia).
false.

5 ?- listens2Music(mia).
true.

6 ?- listens2Music(yolanda).
true.

7 ?-  happy(X).
X = yolanda.

8 ?- playsAirGuitar(Y).
Y = mia ;
Y = yolanda.

9 ?-
```

8. Post Experimental Exercise

A. Questions:

1. Define the terms-> (i) atom (ii) variable. Give examples

(i) atom:- An atom is a general-purpose name with no inherent meaning. It is composed of a sequence of characters that is parsed by the Prolog reader as a single unit. Atoms are usually bare words in Prolog code, written with no special syntax. However, atoms containing spaces or certain other special characters must be surrounded by single quotes. Atoms beginning with a capital letter must also be quoted, to distinguish them from variables. The empty list, written [], is also an atom. Examples of atoms include x, blue, 'Taco', and 'some atom', ++++.

(ii) variable :- Variables are denoted by a string consisting of letters, numbers and underscore characters, and beginning with an upper-case letter or underscore. Variables closely resemble variables in logic in that they are placeholders for arbitrary terms. A variable can become instantiated (bound to equal a specific term) via unification. A single underscore (_) denotes an anonymous variable and means "any term". Unlike other variables, the underscore does not represent the same value everywhere it occurs within a predicate definition.

Example of variable include X, Y, Author, Cat, Dog, My_Colour

2. For the above two knowledge, make a table to enlist the English meaning of the facts and rules

Fact/Rule	English Meaning
woman(mia).	Mia is a woman.
woman(jody).	Jody is a woman.
woman(yolanda).	Yolanda is a woman.
playsAirGuitar(jody).	Jody plays air guitar.
happy(yolanda).	Yolanda is happy.
listens2Music(mia).	Mia listens to music.
listens2Music(yolanda):- happy(yolanda).	If Yolanda is happy she listens to music.
playsAirGuitar(mia):- listens2Music(mia).	If Mia listens to music she plays air guitar.
playsAirGuitar(yolanda):-listens2Music(yolanda).	If Yolanda listens to music she plays air guitar.

3. Explain the significance of the operators ‘,’ and ‘:-‘

The comma ‘,’ character in Prolog is used to signify “and”, whereas the ‘:-’ character is used to signify implication and is read as “if”.

Example:

```
dog(fido).
barks(fido).
isAngry(X):-dog(X),barks(X).
```

In the first line we said that Fido is a dog. In the second line we said that Fido barks. In the third line the part before ‘:-’ is called the head of the clause, ‘:-’ is the neck of the clause and part following ‘:-’ is called body of the clause. The third line can be read as “ X is angry if X is a dog and X barks (for all X)”. The third line is a rule, and if the body of the rule is true the head is also true. If we fire a query like “isAngry(fido).” . We get an output as true, because fido is a dog and he barks, hence fido is angry.

4. WAP to create knowledge base

loves(vincent,mia).
loves(marsellus,mia).
loves(pumpkin,honey_bunny).
loves(honey_bunny,pumpkin).
jealous(X,Y):- loves(X,Z), loves(Y,Z). Also generate queries to get results

```
1 ?- cd('D:/College/PCPF/Prolog/Experiments/Exp10').
true.

2 ?- consult('Q4.pl').
true.

3 ?- loves(vincent,mia).
true.

4 ?- loves(marsellus,mia).
true.

5 ?- loves(marsellus,jody).
false.

6 ?- loves(jade,mia).
false.

7 ?- loves(vincent,X).
X = mia.

8 ?- loves(marsellus,X).
X = mia.

9 ?- loves(X,mia).
X = vincent ;
X = marsellus.

10 ?-
```

```
10 ?- loves(X,Y).
X = vincent,
Y = mia ;
X = marsellus,
Y = mia ;
X = pumpkin,
Y = honey_bunny ;
X = honey_bunny,
Y = pumpkin.

11 ?- jealous(vincent,marsellus).
true.

12 ?- jealous(marsellus,vincent).
true .

13 ?- jealous(vincent,X).
X = vincent ;
X = marsellus.

14 ?- jealous(marsellus,X).
X = vincent ;
X = marsellus.
```

```
15 ?- jealous(X,Y).
X = Y, Y = vincent ;
X = vincent,
Y = marsellus ;
X = marsellus,
Y = vincent ;
X = Y, Y = marsellus ;
X = Y, Y = pumpkin ;
X = Y, Y = honey_bunny.

16 ?- loves(pumpkin,honey_bunny).
true.

17 ?- loves(honey_bunny,pumpkin).
true.

18 ?-
```

C. Conclusion:

In this experiment we have studied how facts, rules and knowledge bases are implemented in Prolog. We have also studied the general syntax of facts and rules, also how to find what is the english meaning of any particular fact or rule. We have executed various queries to analyze the outputs of the programs.

To perform this experiment we have used the SWI-Prolog version 8.2.1 console and editor.

Prolog programs describe relations, defined by means of clauses. There are two types of clauses: Facts and rules. A rule is of the form 'Head :- Body.' and is read as "Head is true if Body is true". Clauses with empty bodies are called facts. An example of a fact is: 'cat(tom).' which is equivalent to the rule: 'cat(tom) :- true.'.

D. References:

[1] Michael L Scott, "Programming Language Pragmatics", Third edition, Elsevier publication
[2] Max Bramer, " Logic Programming with Prolog", Springer, 2005