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

Experiment-8

1. **Aim: a)** To implement recursion functions in prolog
   b) To perform database manipulations using Prolog

2. **Objectives:** After performing the experiment, the students will be able to
   - Understand and implement recursion
   - Perform simple database manipulation operations

3. **Lab Objective Mapped:** To understand, formulate and implement declarative programming paradigm through logic programming

4. **Prerequisite:** Knowledge of facts, rules, constants and variables

5. **Requirements:** The following are the requirements – • **Internet connection**
   - **Laptop/desktop with Windows/Linux/MAC operating system**
   - **SWI Prolog**

6. **Pre-Experiment Theory:**

Recursive Programming

Most programming languages provide loops that allow a set of instructions to be executed repeatedly either a fixed number of times or until a given condition is met. Prolog has no looping facilities. The same effects can be obtained (that enable a sequence of events to be evaluated repeatedly) through backtracking, recursion, built in predicates or a combination of both. A recursive definition in Prolog always has at-least two parts- a first fact that acts like a stopping condition and a rule that calls itself simplified. At each level the first fact is checked. If the fact is true then the recursion ends. If not the recursion continues. A recursive rule must never call itself with same arguments. If that happens the program will never end. When writing such recursive programs in Prolog, a standard recursive pattern always has at least two parts:

Base (non-recursive) clause: Typically the base-case rule(s) will represent the smallest possible example(s) of the problem that you are trying to solve - a list with no members, or just one member. It non-recursively describes the base of the recursive process.

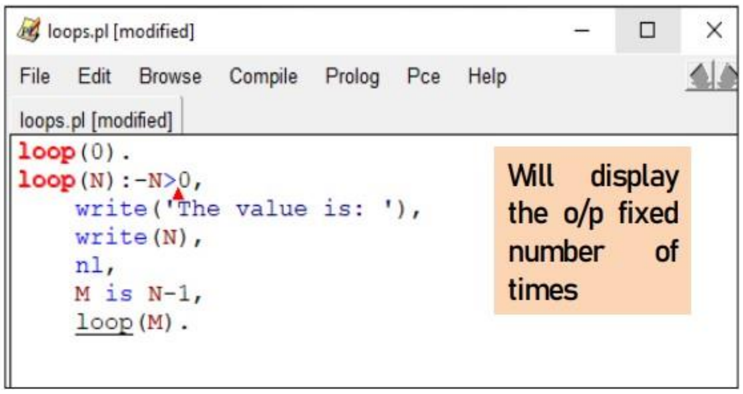
Recursive (continuing) clause: Contains any required logic including a call to itself, continuing recursion.

In prolog, a clause can call itself recursively.

Example: To display n natural numbers recursively in descending order.

The first line of the code defines the non-recursive (base) clause. For the base case the outputs are always pre-defined. In this case, the clause defines to stop for the value of N=0. The base case is preceded by a dot operator.

The loop definition starts from the second line, and continues to call it itself recursively, till the base clause is satisfied (true).


loops.pl [modified]
File Edit Browse Compile Prolog Pce Help
loops.pl [modified]
Will display the o/p fixed number of times

```
loop(0).
loop(N):-N>0,
    write('The value is: '),
    write(N),
    nl,
    M is N-1,
    loop(M).
```

Fig 1: Recursive program to display n natural numbers

Database Manipulation

A Prolog program can be viewed as a database where the specification of relations between data is partly explicit (facts) and partly implicit (rules). Sometimes we want to be able to manipulate this database during execution (e.g. update facts, deduce new rules, etc). Prolog has four database manipulation commands: *assert, retract, asserta*, and *assertz*. The function of the basic database manipulation commands are listed in table 1.

Table 1: Database manipulation commands

Sr. No	**Prolog Command**	**Function**	**Example**
1.	listing	Provides the associated facts of database	?- listing.
2.	assert	To add more facts to the listing	?-assert(son(tom,sue)).
3.	retract	To remove facts from the database when we no longer need them.	?- retract(happy(marcellus)).
4.	assertz.	To place asserted material at the *end* of the database.	assert(p(b)), assertz(p(c)), asserta(p(a)).
5.	asserta.	To place asserted material at the *beginning* of the database.	

7. Lab Laboratory Exercise:
A. Procedure

Steps to be implemented (For recursive)

1. Open SWI-Prolog
2. Go to ….File-> new
3. New prolog editor will open up.
4. Write your program (collection of facts, rules, clauses)

5. Save the file at the desired location as ‘abc.pl’ file
6. Follow the steps -> Save buffer, Make and Compile buffer
7. After successful compilation, at the prompt, first change to working directory using cd command Eg. cd('D:/PCPF/AY-2021-22/Course Lab/Prolog Codes').
8. Load the required file Eg. [abc]
9. To check the outputs, fire appropriate queries at the prompt
10. For database commands execute at the Prolog prompt.

B. Program Code

1. WAP in Prolog to to calculate factorial of a number using recursion
2. WAP in Prolog to find the sum of first n natural numbers using recursion
3. Create a knowledge base1 to include few facts and rules. On this knowledge base, using database manipulation command- (a) append facts (b) append rules (c) delete facts (d) delete rule (e) append facts at the beginning and end of knowledge base

8. Post Experimental Exercise-

A. Questions:

1. Write a program in Prolog to print fibonacci series of number N using recursion
2. Create a knowledge base consisting of the following facts happy(mia). happy(vincent). happy(marsellus). happy(butch).
   happy(vincent).
   To this knowledge base append the following facts happy(jia) at the beginning and happy(john) at the end. Also delete the fact happy(marsellus).
3. For given program how would prolog respond to the query if you keep entering ';' after each solution?
   Program: p(a,b). p(b,c).
   p(X, Y):-p(Y, X).
   Query :
   ?- p(X,Y).

B. Results/Observations/Program output:

Present the program input/output results if any and comment on the same.

C. Conclusion:

1. Write what was performed in the experiment
2. Write which tools you used to perform the experiment
3. Write what you inferred from the output obtained

D. References:

[1] Michael L Scott, “Programming Language Pragmatics”, Third edition, Elsevier publication
[2] Max Bramer, “ Logic Programming with Prolog”, Springer, 2005
[3] https://www.youtube.com/watch?v=iJhtgWAGUAQ [Lecture 14 Prolog Programming]