KATHFORD
INTERNATIONAL COLLEGE OF ENGINEERING & MANAGEMENT || AFFILIATED TO TRIBHUVAN UNIVERSITY

Practical Report: 1

Submitted by:

Sakar KC

BEI IV/I

Roll no.: 08

Subject: Artificial Intelligence

Submitted To:

Department of Electronics, Communication and Information Engineering

Date: July, 2079

LAB 1: INTRODUCTION TO PROLOG

THEORY:

Prolog is a programming language for non-numeric computation. It is well suited for problems involving objects and relationship between objects. In prolog, users develop the problems using facts and rules and define a goal. Prolog then attempts to prove this goal by using formal reasoning on the given data (rules and facts).

Structure of prolog program:

It consists of 4 parts:

1. Domain: A computer always stores a limited knowledge in comparing with the human brains. In any expert system knowledge to stored generally for only a single subjects area known as domain. In prolog programming the domain defines the data type definition such as integer, symbol, real, char, etc.

2. Predicates: It is a function with a value of true or false. Predicates express a relationship.

3. Clauses: The expression in prolog is called clause. In prolog, a clause is terminated with the period(.).

4. Goal: A goal is essentially a question. If prolog can match with fact in the database, it success and respond with TRUE. Each time when a goal is specified, one of the following three results will occurs.

a) The goal will succeed, i.e., it will prove TRUE.

b) The goal will fail, i.e., the prolog will not be ale to match the goals with any facts in program.

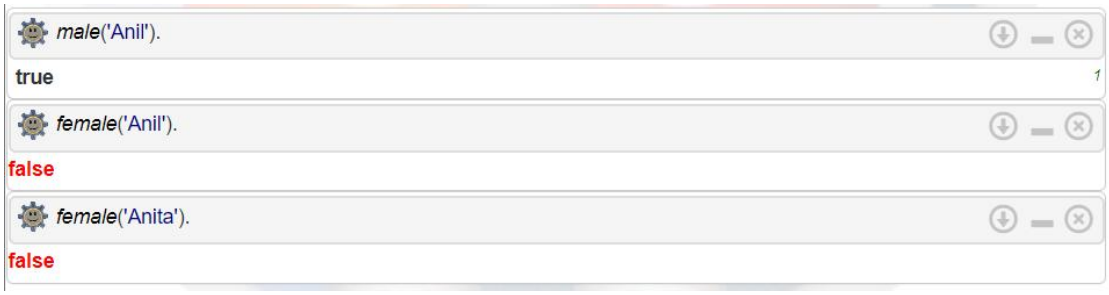
c) The execution will fail because of an error in programming.

OBSERVATION:

Source code:

```
male('Amrit').
male('Anil').
female('Amala').
female('Angana').
```

Queries with result:



Discussion:

In this lab, we learned the programming language called 'Prolog.' Clauses were created and saved on file with the '*.pl' extension. Then, prolog software was opened, and the saved file was consulted. Finally, various outputs were observed by writing queries.

CONCLUSION:

After a brief acquaintance with 'prolog' programming language, the lab was helped to understand how clauses can be used by prolog to prove a goal by using formal reasoning on the given data.

LAB 2: FAMILIARIZATION WITH PROLOG PREDICATES

Theory:

Some of the prolog predicates are listed below.

Name	Predicate Calculus	Prolog Symbol
AND	∩	,
OR	V	;
Only if	←	:-
NOT	¬	Not

1. AND: It is used for that case where one or more statement should be true.
2. OR: It return time if one of the statement in the predicate is true
3. Only if: It implies that the left statement will be executed following the statement (s) after the symbol ':-'.
4. Not: Built-in NOT predicate can only be used in premise.

OBSERVATION:

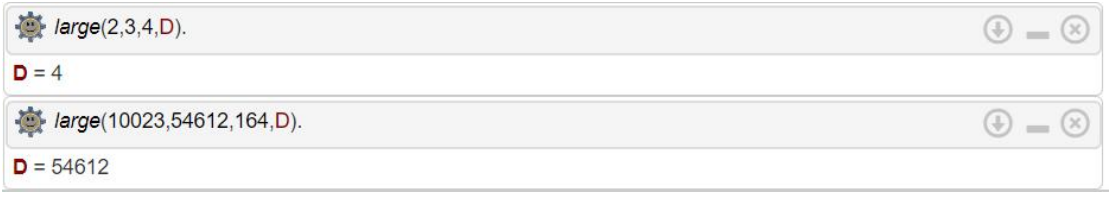
Source code 1:

```
large(A,B,C,D):-A>B, A>C, D=A.
large(A,B,C,D):-B>A, B>C, D=B.
large(A,B,C,D):-C>A, C>B, D=C.
```

Queries:

Largest among 2,3,4 is?

Output:



Source code 2:

```
sum(X,Y):-S is X+Y, write(S).
```

Output:

Source code 3:

```
food('burger').
food('sandwich').
food('pizza').
lunch('sandwich').
dinner('pizza').
meal(X):-food(X).
```

Queries:

1. Is pizza a food?
2. Which food is meal and lunch?
3. Is sandwich a dinner?

Output:

Source code 4:

```
student('Hari').
student('Ram').
student('Bhim').
student('Sakar').
student('Aashish').
takesMorningClass('Bhim').
takesMorningClass('Sakar').
takesEveningClass('Hari').
takesEveningClass('Ram').
canTakeExam(X):-student(X),takesMorningClass(X).
```

Queries:

Who can take exam?

Output:

Source code 5:

```
owns('Jack',car(bmw)).
owns('John', car(chevy)).
owns('Olivia', car(civic)).
owns('Jane', car(chevy)).
sedan(car(bmw)).
sedan(car(civic)).
truck(car(chevy)).
```

Queries:

1. What does john own?
2. Does John own something?
3. Who owns car chevy?
4. Does Jane own sedan?
5. Does Jane own truck?

Output:

Discussion:

In this lab, we learned the Prolog predicate. For this lab, I have used this website: https://swish.swi-prolog.org/p/Hangout.swinb.
Clauses were written, and according to the problem, queries were written, and outputs were tallied to the answer. Thus, with clear rules, clauses, and queries, logic can be implemented in prolog language efficiently.

CONCLUSION:

After a brief study of predicates of 'prolog' programming language and website: swi-prolog.org, a comprehensive understanding of logic and its implementation via programming language developed.

LAB 3: IMPLEMENTATION OF BACKTRACKING IN PROLOG

THEORY:

It is a general algorithm for finding all or some solutions to some computational problems, notably constraints satisfaction problems, that incrementally builds candidates to the solutions, and abandons each partial candidate C ("backtracks") as soon as it determines that C cannot possibly be completed to a valid solution.

OBSERVATION:

Source Code 1:

```
% clauses:
like(mary, food).
like(mary, wine).
like(hari, beer).
like(sakar, milk).
like(hari,X):-like(mary,X).
food(burger).
food(sandwich).
food(pizza).
lunch(sandwich).
dinner(pizza).
meal(X):-food(X).
```

Output with queries:

Source Code 2:

```
% clauses:
studies(gopal, course(135)).
studies(ram, course(135)).
studies(hari, course(135)).
studies(shyam, course(131)).
teaches(kiran, course(135)).
teaches(bipin, course(131)).
teaches(amrit, course(171)).
professor(X,Y):-teaches(X,C),studies(Y,C).
```

Output:

DISCUSSION:

In this lab, Backtracking in the Prolog programming language were implemented. For this lab, I have used this website: https://swish.swi-prolog.org/p/Hangout.swinb. Clauses were written, and according to the problem. Then, queries were written, and outputs were tallied to the answer. Thus, with clear rules, clauses, and queries, logic can be implemented in prolog language efficiently.

CONCLUSION:

After a brief study of backtracking in prolog and website: swi-prolog.org, a comprehensive understanding of logic and its implementation via programming language developed.

LAB 4: IMPLEMENTATION OF DIFFERENT ARITHMETIC OPERATIONS IN PROLOG

THEORY:

A special predefined operator like 'is', perform arithmetic operations.
For example:

?- X is 7+2

The answer will be:

X=9

It is important to note that expressions is not evaluated by itself. We have to suply a variable to collect a result. Some pre-defined prolog arithmetic infix operators are:

>, greater than
<, less than
>=, greater than equal to
=<, less than equal to

OBSERVATION:

Program 1: Factorial of a number

Source code:

```
fact(0,1).
fact(N,F):-
    N>0,
    N1 is N-1,
    fact(N1,F1),
    F is N*F1.
```

Output:

Program 2: power of a number

Source Code:

```
power(0,N,0):- N>0.
power(X,0,1):- X>0.
power(X,N,V):-X>0,N>0,N1 is N-1,power(X,N1,V1), V is V1*X.
```

Output:

DISCUSSION:

In this lab, different arithmetic operations like factorial and power of a number, were implemented in the Prolog programming language. For this lab, I have used this website: https://swish.swi-prolog.org/p/Hangout.swinb.
Clauses were written, and according to the problem. Then, queries were written, and outputs were tallied to the answer. Thus, with clear rules, clauses, and queries, logic can be implemented in prolog language efficiently.

CONCLUSION:

After a brief study of different arithmetic operations in prolog and website: swi-prolog.org, a comprehensive understanding of logic and its implementation via programming language developed.