PART A

Experiment No.05

- **Aim:** Case study on Basic Programming in PROLOG and Develop a program to implement family tree.
- **Prerequisite:** Understand knowledge, facts and rules

- **Outcome:**

After successful completion of this experiment students will be able to

- Use current techniques, tools necessary for computing practice using PROLOG.
- Applying knowledge of computing, applied mathematics to solve engineering problems using logical programming.
- Ability to recognize the need and to engage in lifelong learning for logical programming.

Tools Required: SWI-Prolog

- **Theory:**

Prolog is a general-purpose logic programming language associated with artificial intelligence and computational linguistics. Prolog has its roots in first-order logic, a formal logic, and unlike many other programming languages, Prolog is declarative: the program logic is expressed in terms of relations, represented as facts and rules. A computation is initiated by running a query over these relations. Prolog is well-suited for specific tasks that benefit from rule- based logical queries such as searching databases, voice control systems, and filling templates.

Rules and facts:

Prolog programs describe relations, defined by means of clauses. Pure Prolog is restricted to Horn 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". A rule's body consists of calls to predicates, which are called the rule's goals. The built-in predicate, /2 (meaning a 2-arity operator with name,) denotes conjunction of goals, and; /2 denotes disjunction. Conjunctions and disjunctions can only appear in the body, not in the head of a rule.

Clauses with empty bodies are called facts. An example of

a fact is: cat (tom). which is equivalent to the rule:

cat(tom) :- true.

The built-in predicate true/0 is

always true. Given the above fact,

one can ask:

```
  is tom a cat?
 ?- cat(tom).
   Yes
what things are cats?
?- cat(X).

X = tom
```

Clauses with bodies are called rules. An example
of a rule is: animal(X) :- cat(X).

If we add that rule and ask what things are animals?

```
?-
an
i
m
al
(
X
).
X
=
to
m
```

Prolog must be able to handle arithmetic in order to be a useful general purpose programming language. However, arithmetic does not fit nicely into the logical scheme of things. That is, the concept of evaluating an arithmetic expression is in contrast to the straight pattern matching we have seen so far. For this reason, Prolog provides the built-in predicate 'is' that evaluates arithmetic expressions. Its syntax calls for the use of operators.

X is <arithmetic expression>

The variable X is set to the value of the arithmetic expression. On backtracking it is unassigned. The variable X is set to the value of the arithmetic expression. On backtracking it is unassigned. The arithmetic expression looks like an arithmetic expression in any other programming language. Here is how to use Prolog as a calculator.

```
?- X is 2 + 2.
X = 4
?- X is 3 * 4 + 2.
X = 14
```

Parentheses clarify precedence.

```
?- X is 3 * (4 + 2).
X = 18
?- X is (8 / 4) / 2.
X = 1
```

In addition to 'is,' Prolog provides a number of operators that compare two numbers. These include 'greater than', 'less than', 'greater or equal than', and 'less or equal than.' They behave more logically, and succeed or fail according to whether the comparison is true or false. Notice the order of the symbols in the greater or equal than and less than or equal operators. They are specifically constructed not to look like an arrow, so that the use arrow symbols in programs is without confusion.

X

>

Y

X

<

Y

X

>

=

Y

X

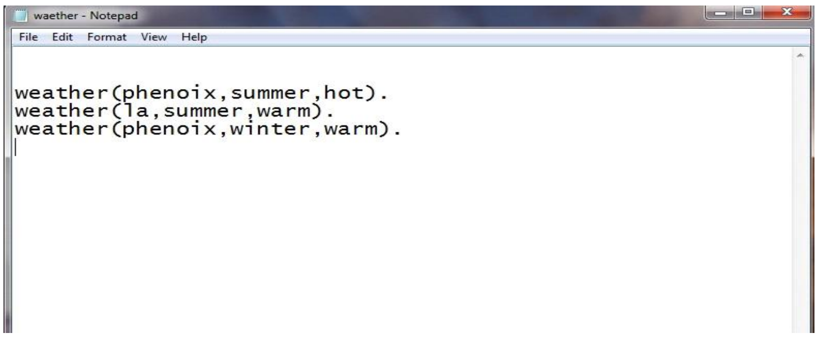
=

<

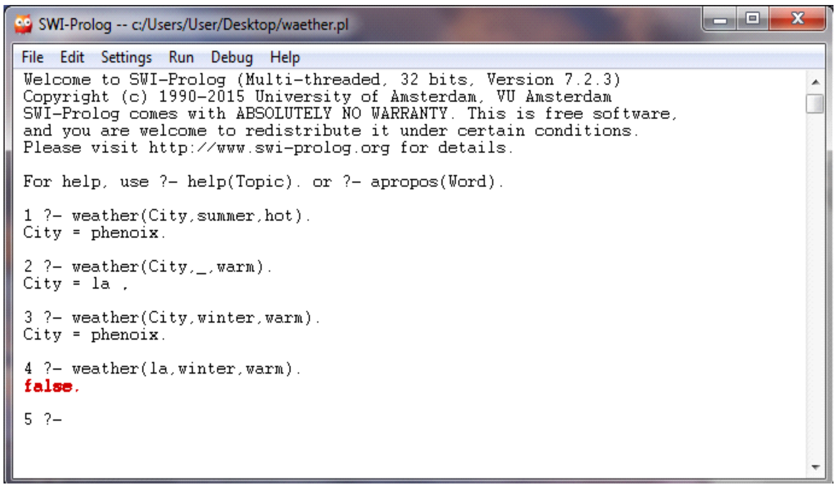
Y

- **Procedure/ Program:**

1 (A). Sample program to demonstrate Rules and facts.

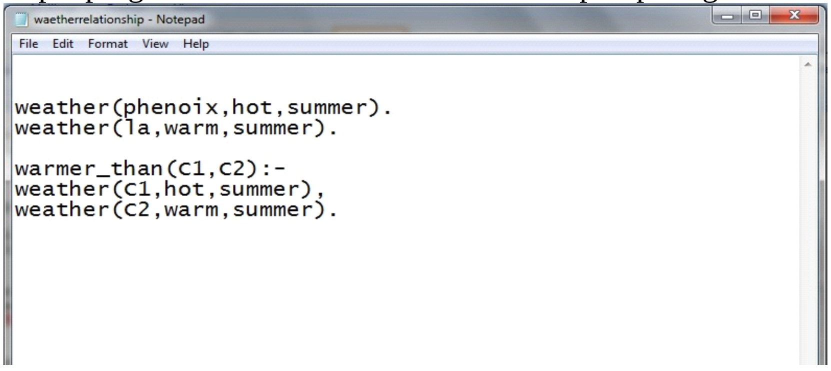


```
weather(phenoix,summer,hot).
weather(la,summer,warm).
weather(phenoix,winter,warm).
```

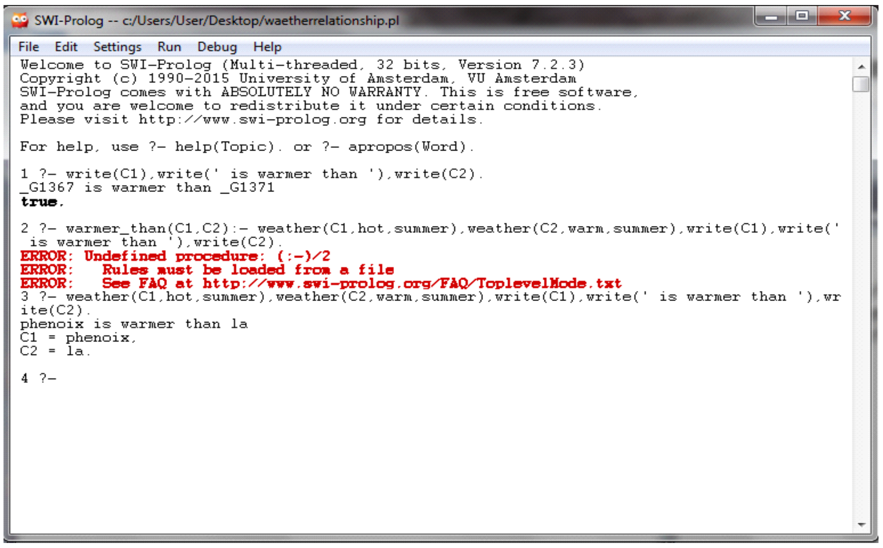

SWI-Prolog -- c:/Users/User/Desktop/waether.pl
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- weather(City,summer,hot).
City = phenoix.

2 ?- weather(City,_,warm).
City = la ,

3 ?- weather(City,winter,warm).
City = phenoix.

4 ?- weather(la,winter,warm).
false.

5 ?-
```

1.(B) Sample program to demonstrate the relationship in prolog.

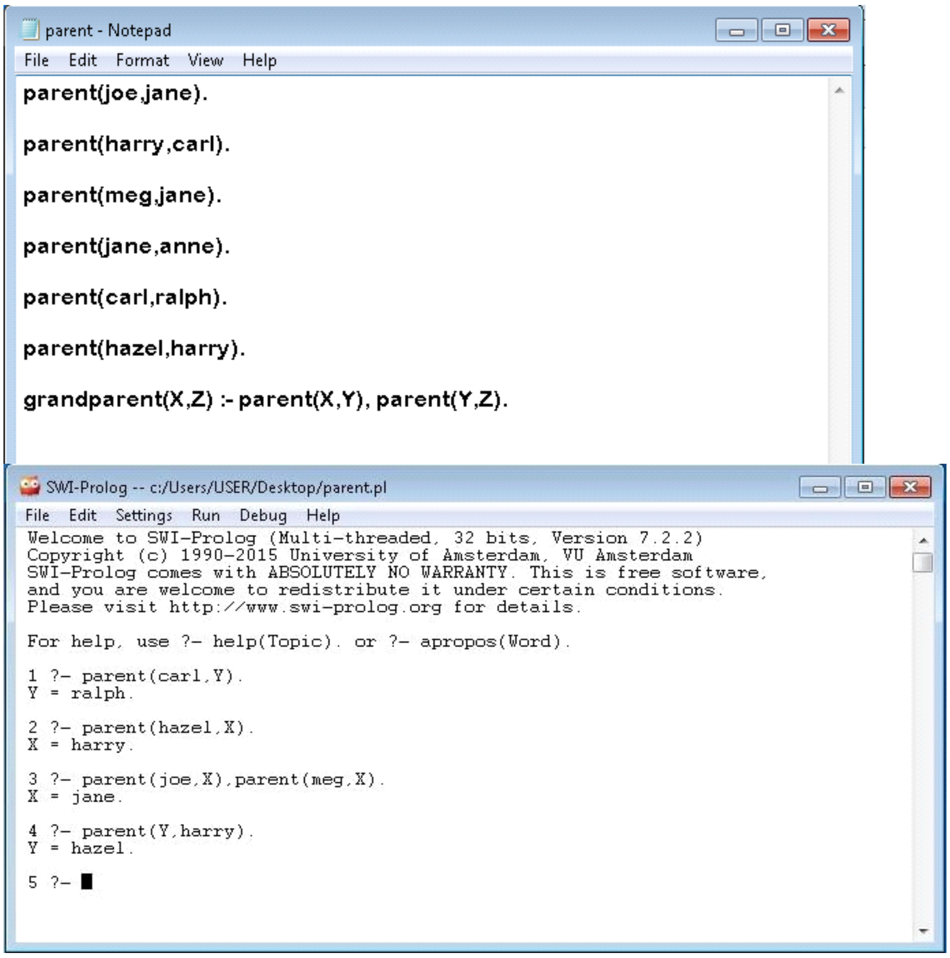


```
weather(phenoix,hot,summer).
weather(la,warm,summer).

warmer_than(C1,C2):-
weather(C1,hot,summer),
weather(C2,warm,summer).
```


SWI-Prolog -- c:/Users/User/Desktop/waetherrelationship.pl
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 7.2.3)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- write(C1),write(' is warmer than '),write(C2).
_G1367 is warmer than _G1371
true.

2 ?- warmer_than(C1,C2):- weather(C1,hot,summer),weather(C2,warm,summer),write(C1),write('
 is warmer than '),write(C2).
ERROR: Undefined procedure: (:-)/2
ERROR:   Rules must be loaded from a file
ERROR:   See FAQ at http://www.swi-prolog.org/FAQ/ToplevelMode.txt
3 ?- weather(C1,hot,summer),weather(C2,warm,summer),write(C1),write(' is warmer than '),wr
ite(C2).
phenoix is warmer than la
C1 = phenoix,
C2 = la.

4 ?-
```

1.(C) Sample program to demonstrate the relationship in prolog.


parent - Notepad
File Edit Format View Help
SWI-Prolog -- c:/Users/USER/Desktop/parent.pl
File Edit Settings Run Debug Help

```
parent(joe,jane).

parent(harry,carl).

parent(meg,jane).

parent(jane,anne).

parent(carl,ralph).

parent(hazel,harry).

grandparent(X,Z) :- parent(X,Y), parent(Y,Z).
```

```
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 7.2.2)
Copyright (c) 1990-2015 University of Amsterdam, VU Amsterdam
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to redistribute it under certain conditions.
Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

1 ?- parent(carl,Y).
Y = ralph.

2 ?- parent(hazel,X).
X = harry.

3 ?- parent(joe,X),parent(meg,X).
X = jane.

4 ?- parent(Y,harry).
Y = hazel.

5 ?- 
```

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

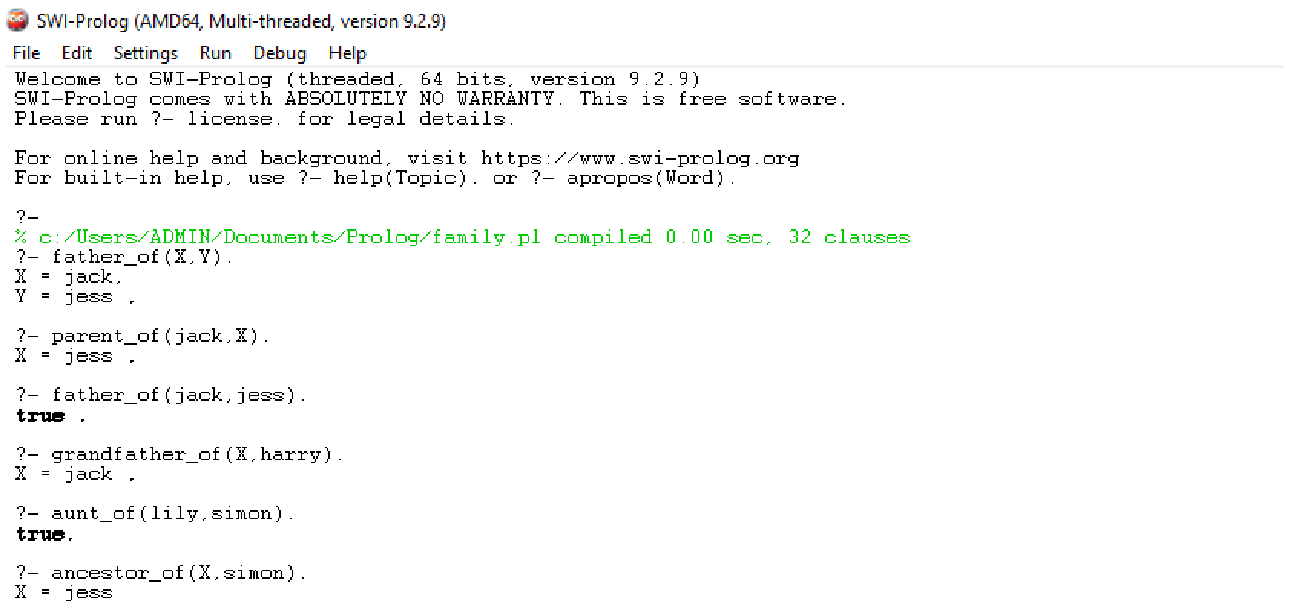
Roll. No. B30	Name : Pranjal Bhatt
Class : TE-COMPS B	Batch : B2
Date of Experiment :	Date of Submission:
Grade:	

- **Software Code written by student:**

```
/* 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).

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).
```

- **Input and Output:**


SWI-Prolog (AMD64, Multi-threaded, version 9.2.9)
File Edit Settings Run Debug Help

```
Welcome to SWI-Prolog (threaded, 64 bits, version 9.2.9)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?-
% c:/Users/ADMIN/Documents/Prolog/family.pl compiled 0.00 sec, 32 clauses
?- father_of(X,Y).
X = jack,
Y = jess .

?- parent_of(jack,X).
X = jess .

?- father_of(jack,jess).
true .

?- grandfather_of(X,harry).
X = jack .

?- aunt_of(lily,simon).
true.

?- ancestor_of(X,simon).
X = jess
```

- **Observations and learning:**

We understood that in prolog relation, the specification is between the objects and the properties of the object, as we have seen in this article we can make relationship easily by making family trees and illustrating them, in prolog program the family trees can be illustrated by taking facts, rules and by posing queries

- **Conclusion:**

In this experiment, we aimed to set up and execute a Prolog program using SWI-Prolog. Initially, the swipl command was not recognized in the command prompt, which indicated a possible issue with installation or system configuration. Various troubleshooting methods were applied, such as checking if Prolog was installed in the correct directory, running the executable directly, and verifying if it was listed in the system's environment variables.We understood the family tree .

- **Question of Curiosity:**
  - Consider the following bachelor Prolog program.

  What would it be the "INCORRECT" result of the following query?

```
bachelor(P) :- male(P), not married(P).
male(henry).
male(tom).
married(tom).

?- bachelor(henry).
% yes

?- bachelor(tom).
```

```
% no

?- bachelor(Who).
% Who = henry

?- married(X).
% X = tom

?- male(P).
% no
```

Answer : E. ?- male(P).

- Which of the following is *not* a query? (I.e., which of the following does not conform to the syntax of queries?)

● ○ ?- student(Lisa, 5).

○ ?- student(Lisa, X),
student(Abraham, X). ○ ?-
student(Abraham, X)

Answer : ○ ?- student(Lisa, 5).