Lab Record

of

Artificial Intelligence

(CSF304)

Submitted to:

Ankit Agarwal
Assistant Professor
School of Computing

Submitted by:

SATYAM RAJ
Sap id: 1000015607
Section: I (P1)

Session 2022-23

Index

S.No.	Title of Experiment/Objective	Date of Conduction	Signature of Faculty
1	Introduction to PROLOG programming, PROLOG platform. “Hello World” program.		
2	Defining Clauses and Predicates, Variables, Anonymous Variables.		
3	Arithmetic Operators, Arithmetic Functions and Logical Operators (NOT, conjunction disjunction).		
4	Binding Variables and Backtracking & Concept of Unification.		
5	Implementation of Recursion in PROLOG.		
6	Implementation of LIST and built-in predicates of LIST in PROLOG.		
7	Implementation of State-Space Searching Problem using PROLOG (Water-Jug or 8 Queens problem).		
8	Universal and Existential Quantifier Variables in PROLOG.		
9	Knowledge Base and Rule Base Creation for a specific domain in PROLOG.		
10	Implementation of Resolution process in PROLOG.		
11	Implementation of an Expert System for a particular domain.		

Practical -1

Introduction to PROLOG programming, PROLOG platform.

"Hello World" program.

PROLOG-PROGRAMMING IN LOGIC

PROLOG stands for Programming, In Logic — an idea that emerged in the early 1970's to use logic as programming language. The early developers of this idea included Robert Kowaiski at Edinburgh (on the theoretical side), Marrten van Emden at Edinburgh (experimental demonstration) and Alian Colmerauer at Marseilles (implementation). David D.H. Warren's efficient implementation at Edinburgh in the mid -1970's greatly contributed to the popularity of PROLOG.

PROLOG is a programming language centered on a small set of basic mechanisms, including pattern matching, tree-based data structuring and automatic backtracking. This Small set constitutes a surprisingly powerful and flexible programming framework. PROLOG is especially well suited for problems that involve objects- in particular, structured objects- and relations between them.

SYMBOLIC LANGUAGE

PROLOG is a programming language for symbolic, non-numeric computation. It is especially well suited for solving problems that involve objects and relations between objects.

For example, it is an easy exercise in prolog to express spatial relationship between objects, such as the blue sphere is behind the green one. It is also easy to state a more general rule: if object X is closer to the observer than object Y. and object Y is closer than Z, then X must be closer than Z. PROLOG can reason about the spatial relationships and their consistency with respect to the general rule. Features like this make PROLOG a powerful language for Artificial Intelligence (AI) and non- numerical programming.

There are well-known examples of symbolic computation whose implementation in other standard languages took tens of pages of indigestible code, when the same algorithms were implemented in PROLOG, the result was a crystal-clear program easily fitting on one page.

Print Hello World using message() and write() function in Prolog.

Solution: -

USING WRITE () FUNCTION

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

?- write('hello World').
hello World
true.
```

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

?- write('Lab experiment:1').
Lab experiment:1
true.

?- write('Artificial Intelligence').
Artificial Intelligence
true.

?- X is 1,write(X).
1
X = 1.
```

Practical -2

Aim: Write a prolog program to form clauses and predicates by the help of variables and anonymous variables.

Solution: -

Programming in PROIOG is accomplished by creating a database of facts and rules about objects, their properties, and their relationships to other objects.

a. FACTS: Some facts about family relationships could be written as:
sister(sue,bill)
parent(ann.sam)
male(jo) female(riya)

b. RULES: To represent the general rule for grandfather, write:
grandfather(X,Z)
parent(X,Y) parent(
Y,Z) male(X)

c. QUERIES: Given a database of facts and rules such as that above, execute following statements: ?-
parent(X,sam)
X=ann
?grandfather(X,Y)
X=jo, Y=sam

File Edit View

```
sister(sue,bill).
parent(ann,sam).
male(jo).
female(riya).
grandfather(X,Z).
 parent(X,Y).
parent(Y,Z).
male(X).
```

```
% c:/Users/dit/Desktop/ok.pl compiled 0.00 sec, 7 clauses
?- parent(ann,Y).
Y = sam ,

?- grandfather(X,Y).
true.

?- parent(sue,Y).
true .

?- parent(sue,Z).
true
```

```
parent(pam,bob).
parent(tom,bob).
parent(tom,liz).
parent(bob,ann).
parent(bob,pat).
parent(pat,jim).
```

```
% c:/Users/sonal mishra/OneDrive/Desktop/5th semester/ai lab/notepad/lab1.pl compiled 0.00 sec, 6 clauses
?-
|    parent(pam,liz).
false.

?- parent(X,pat).
X = bob.

?- parent(tom,X).
X = bob ;

?- parent(X,ann).
X = bob.

?- parent(X,jim).
X = pat.

?- parent(tom,jim).
false.
```

```
father(d,i).
father(h,m).
father(f,k).
father(k,n).
father(k,o).
father(a,c,e,f).
mother(b,e).
mother(b,f).
mother(c,i).
mother(i,m).
mother(g,k).
mother(i,n).
mother(i,o).
mother(b,c,e,f).
grandp(m,d,c).
grandp(i,a,b).
grandp(k,a,b).
grandp(n,f,g).
grandp(o,f,g).
ife(a,b).
wife(d,c).
wife(f,g).
wife(h,i).
wife(k,i).
brother(o,n).
brother(c,f).
```

```
?-
% c:/Users/sonal mishra/OneDrive/Desktop/ok.pl compiled 0.00 sec, 29 clauses
?- father(d,X).
X = i.

?- father(Y,k).
Y = f.

?- father(a,c,e,X).
X = f.

?- father(k,Y).
Y = n ;

?- mother(b,Y).
Y = e ;

?- mother(i,X).
X = m ;

?- mother(b,c,Y,f).
Y = e.
```

Practical-3

Aim: Write a prolog program for arithmetic operators, arithmetic functions, and logical operators.

Arithmetic Operator: -

Arithmetic Operators in Prolog

Arithmetic operators are used to perform arithmetic operations. There are few different types of arithmetic operators as follows –

Operator	Meaning
+	Addition
-	Subtraction
*	Multiplication
/	Division
**	Power
//	Integer Division
mod	Modulus

```
X is 100 + 200,write('100 + 200 is '),write(X).
Y is 400 - 150,write('400 - 150 is '),write(Y).
Z is 10 * 300,write('10 * 300 is '),write(Z).
A is 100 / 30,write('100 / 30 is '),write(A).
B is 100 // 30,write('100 // 30 is '),write(B).
C is 100 ** 2,write('100 ** 2 is '),write(C).
D is 100 mod 30,write('100 mod 30 is '),write(D).
```

Comparison Operators

Comparison operators are used to compare two equations or states. Following are different comparison operators –

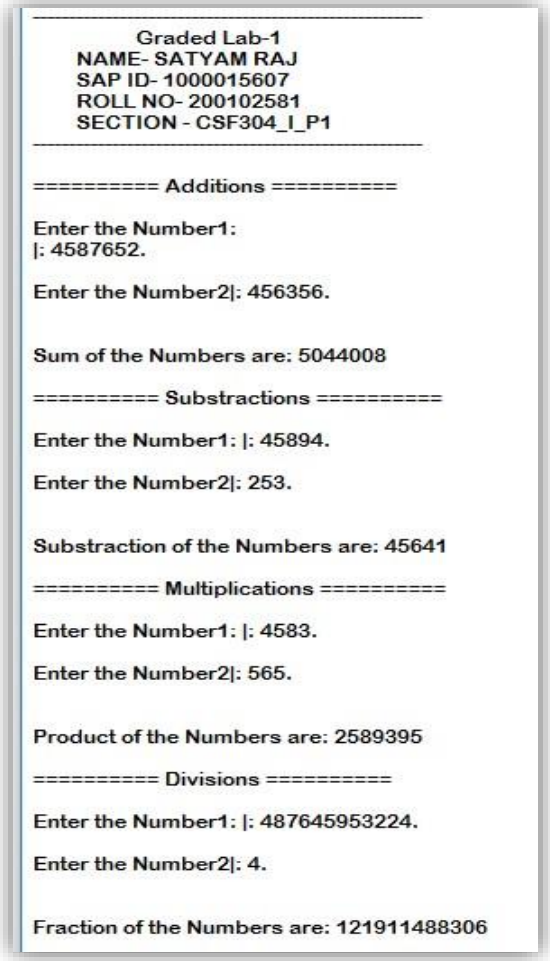
Operator	Meaning
X > Y	X is greater than Y
X < Y	X is less than Y
X >= Y	X is greater than or equal to Y
X =< Y	X is less than or equal to Y
X =:= Y	the X and Y values are equal
X =\= Y	the X and Y values are not equal

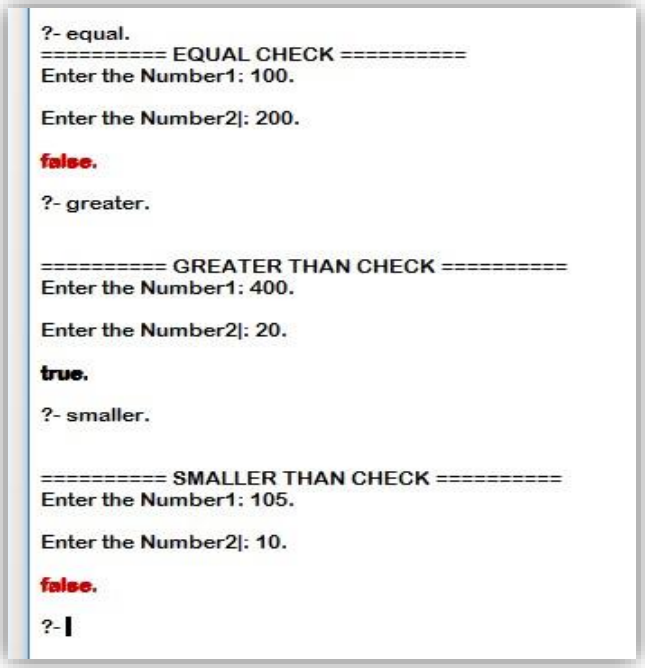
Codes:

```
 gradedlab:-
info,sum,diff,multi,divide.


info:-   write("-------------------------------------------------------
"),nl, write("                      Graded Lab-1                    "),nl,
write(" NAME- SATYAM RAJ"),nl, write(" SAP ID- 1000015607"),nl, write("
ROLL NO- 200102581"),nl, write(" SECTION - CSF304_I_P1"),nl, write("--
-------------------------------------------------"),nl,nl.



write("-------------ARITHMATIC OPERATORS--------------"),nl,nl.
 sum:- write("==========Additions==========="),nl,nl,
write("Enter the Number1: "), read(Num1),nl,
write("Enter the Number2"), read(Num2),nl, Num3 is
Num1+Num2,nl, write("Sum of the Numbers are:
"),write(Num3),nl,nl.

diff:- write("========== Substractions ==========="),nl,nl,
write("Enter the Number1: "), read(Num1),nl, write("Enter the
Number2"), read(Num2),nl, Num3 is Num1-Num2,nl,
write("Substraction of the Numbers are: "),write(Num3),nl,nl.

 multi:- write("========== Multiplications
==========="),nl,nl, write("Enter the Number1: "),
read(Num1),nl, write("Enter the Number2"),
read(Num2),nl,
Num3 is Num1*Num2,nl, write("Product of the Numbers
are:"),write(Num3),nl,nl.

 divide:- write("========== Divisions ==========="),nl,nl,
write("Enter the Number1: "), read(Num1),nl, write("Enter
the Number2"), read(Num2),nl, Num3 is Num1/Num2,nl,
write("Fraction of the Numbers are: "),write(Num3),nl,nl.



%-------------RELATIONAL OPERATORS--------------%
```

```
equal:- write("========== EQUAL CHECK
==========="),nl, write("Enter the Number1: "),
read(Num1),nl, write("Enter the Number2"),
read(Num2),Num1=:=Num2.


greater:- nl,nl,write("========== GREATER THAN CHECK
==========="),nl, write("Enter the Number1: "),
read(Num1),nl, write("Enter the Number2"),
read(Num2),Num1>Num2.

 smaller:-
nl,nl,write("========== SMALLER THAN CHECK ==========="),nl,
write("Enter the Number1: "), read(Num1),nl, write("Enter
the Number2"), read(Num2),Num1<Num2.
```

OUTPUT:





Practical- 4

Aim: Binding Variables and Backtracking & Concept of Unification.

Code:

GL3 - Notepad

File Edit View

```
mother(mia, liam).
mother(mia, haley).
mother(jessica, chris).
mother(jessica, mark).
mother(eva, ben).
mother(haley, jake).
mother(bennet, alicia).
mother(michelle, jennifer).
mother(alicia, chris).
mother(alicia, britt).
father(liam, michael).
father(josh, haley).
father(mark, henry).
father(mark, alicia).
father(josh, chris).
father(taylor, michelle).
father(josh, mark).
father(james, luke).
father(james, miller).
parent(olivia, matt).
parent(olivia, smith).
parent(A, B) :- write('mother?'), nl, mother(A, B), write('mother!'), nl.
parent(X, Y) :- write('father?'), nl, father(X, Y), write('father!'), nl.
parent(alexa, jessa).
parent(alexa, dave).
```

```
7-parent (josh. Child), write( The child is).write(Child).nl.
 Correct to "vrite('The child is )? yes
nother?
father?
father!
The child is haley
Child = haley
```

```
teaches(Teacher, Student):-
    lectures(Teacher, Subject), studies(Student, Subject).
lectures(codd, databases).
studies(fiona, databases).
studies(fred, databases).
```

Output:

```
?-
|     men(bob,robert)=men(bob,robert).
true.

?-
```

```
?-
|     teaches(codd,W).
W = fiona ;

?- teaches(fred,W).
false.

?- teaches(fiona,W).
false.

?- lectures(X,databases).
X = codd.

?- teaches(codd,W).
W = fiona ;
W = fred.

?-
```

Practical- 5

Aim: Implementation of Recursion in PROLOG.

Predicate:

File Edit Format View Help

```
factorial(0, 1).
 factorial(N, F) :-
      N > 0,
      Prev is N -1,
      factorial(Prev, R),
      F is R * N.
```

Output:

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

?-
% c:/Users/dit/Desktop/lab3.pl compiled 0.00 sec, 2 clauses
?- factorial(5,F).
F = 120 ,

?- factorial(3,F).
F = 6 ,

?- factorial(8,F).
F = 40320 ,

?- factorial(4,F).
F = 24 ,

?- factorial(1,F).
F = 1 ,

?- factorial(0,F).
F = 1
```

Practical- 6

Aim: Implementation of list and built-in predicates of LIST in PROLOG

1.Find the number of elements in a list: -

```
list_len([],0).
list_len([_| L], N):-
  list_len(L, N1),
  N is N1 + 1.
```

```
% c:/Users/DITU.DESKTOP-OCL7AEF/Documents/Prolog/lab6.pl compiled 0.00 sec, 8 clause
s
?- %Find the number of elements in a list
|  list_len([x,y,z,a,f,g,t], N).
N = 7.


?- list_len([a,s,d,f,g,h,j,k,l,w,e,r,t],N).
N = 13.
```

2.Find the sum of elements in a list: -

```
list_sum([],0).
list_sum([Head|Tail],Sum):-
    list_sum(Tail,Sum1),
    Sum is Head + Sum1.
```

```
% c:/Users/DITU.DESKTOP-OCL7AEF/Documents/Prolog/lab6.pl compiled 0.00 sec, 0 clause
s
?- list_sum([5,6,9,8,7],Sum).
Sum = 35.

?- list_sum([4,8,5,8,5,8,5,8,6,6,5,9],Sum).
Sum = 77.

?-
```

3.Append: -

% To append the value in the list: -

```
apnd_list([], L2, L2).
apnd_list([X | L1], L2, [X|L3]):-
  apnd_list(L1, L2, L3).
```

?- apnd_list([x,y,x,w],[k,l], Ans).
Ans = [x, y, x, w, k, l].

?- apnd_list([a,s,d,f,g,h,l],[r,t,y,u], Ans).
Ans = [a, s, d, f, g, h, l, r, t|...].

?- apnd_list([a,s,d,f,g],[r,t,y,u], Ans).
Ans = [a, s, d, f, g, r, t, y, u].

4.Member: -

% Find whether the member is present or not

```
is_memb(X ,[X|_]):-!.
is_memb(X ,[_|Last]):-
  is_memb(X,Last).
```

?- is_memb(x,[a,b,c,x,d]).
true.

?- is_memb(z,[a,b,c,x,d]).
false.

?- is_memb(k,[o,p,l,k,s,w]).
true.

?-

Practical- 7

1. Write a program in Prolog to solve '8 Queens problem'.

```
info:- write("-----------------------------------------------------"),nl, write(" Graded Lab-4 "),nl,
write(" NAME-SATYAM RAJ"),nl, write(" SAP ID-1000015607"),nl, write(" ROLL NO-
200102581"),nl, write(" SECTION -CSF304_I_P1"),nl, write("-----------------------------------------------
----"),nl,nl.
commands :- write("List of commands:"), nl, nl, write("print_coordinates"), nl, write("print_board"),
nl, nl. print_coordinates :- solution(A), display_coordinates(A, 1). print_board :- solution(A),
print_boundry, print_columns(A, 1), print_column_letters, nl. print_square :- write("| ").
print_queen :- write("|Q"). print_end_quare :- write("|"), nl. print_boundry :- write("  - - - - - - - - "), nl.

print_column_letters :- write("  h g f e d c b a "), nl.

print_columns([], _).
print_columns([H|T], C) :- write(C),
write(" "), print_row(H, 1), C1 is C + 1,
print_columns(T, C1).

print_row(_, 9) :-
print_end_quare,
print_boundry, !. print_row(X,
Col) :-

X is Col, print_queen,

Col1 is Col + 1, print_row(X, Col1).

print_row(X, Col) :-

X =\= Col, print_square,

Col1 is Col + 1, print_row(X, Col1).

solution(Queens) :- permutation([1,2,3,4,5,6,7,8],
Queens), safe(Queens). permutation([],[]).
permutation([H|T], P) :- permutation(T, P1), mydelete(H,
P, P1).

mydelete(X, [X|L], L).

mydelete(X, [Y|L], [Y|L1]) :- mydelete(X, L, L1).

safe([]).

safe([Queen|Others]) :- safe(Others),
noattack(Queen,Others,1).

noattack(_,[],_). noattack(Y, [Y1|L], X1)
:-
```

```
Y1 - Y =\= X1,

Y - Y1 =\= X1,  X2 is X1 + 1,
noattack(Y, L, X2).

display_coordinates([], 9).

display_coordinates([H|T], Y) :-  nth1(H,
[h,g,f,e,d,c,b,a], X),  write("["), write(X),
write(Y), write("]"), nl,  Y1 is Y + 1,
display_coordinates(T, Y1).
```

OUTPUT:

```
File  Edit  Settings  Run  Debug  Help
Welcome to SWI-Prolog (threaded, 64 bits, version 8.2.4)
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/dit.DESKTOP-F3ALMF1.000/Desktop/stymrj.pl compiled 0.00 sec, 25 clauses
?- info.
-----------------------------------------------------
 Graded Lab-4
 NAME-SATYAM RAJ
 SAP ID-1000015607
 ROLL NO-200102581
 SECTION -CSF304_I_P1
-----------------------------------------------------

true.
```

```
?- commands.
List of commands:

print_coordinates
print_board

true.

?- print_coordinates.
[d1]
[g2]
[c3]
[h4]
[b5]
[e6]
[a7]
[f8]
true ;
[c1]
[f2]
[d3]
[b4]
[h5]
[e6]
[g7]
[a8]
true ;
[c1]
[e2]
[b3]
[h4]
[f5]
[d6]
[g7]
[a8]
true ;
[f1]
[c2]
[g3]
[b4]
[d5]
[h6]
[a7]
[e8]
true ;
[c1]
[f2]
[h3]
[b4]
[d5]
[a6]
[g7]
[e8]
true
```

```
true ;
[c1]
[g2]
[b3]
[h4]
[f5]
[d6]
[a7]
[e8]
true ;
[c1]
[e2]
[b3]
[h4]
[a5]
[g6]
[d7]
[f8]
true ;
[f1]
[c2]
[g3]
[b4]
[h5]
[e6]
[a7]
[d8]
true ;
[c1]
[f2]
[b3]
[g4]
[e5]
[a6]
[h7]
[d8]
true ;
[c1]
[f2]
[b3]
[e4]
[h5]
[a6]
[g7]
[d8]
true ;
[g1]
[c2]
[h3]
[b4]
[e5]
[a6]
[f7]
[d8]
true
```

```
true ;
[c1]
[g2]
[b3]
[h4]
[e5]
[a6]
[d7]
[f8]
true ;
[c1]
[f2]
[b3]
[g4]
[a5]
[d6]
[h7]
[e8]
true ;
[d1]
[b2]
[g3]
[c4]
[f5]
[h6]
[e7]
[a8]
true ;
[d1]
[b2]
[g3]
[c4]
[f5]
[h6]
[a7]
[e8]
true ;
[e1]
[b2]
[d3]
[f4]
[h5]
[c6]
[a7]
[g8]
true ;
[e1]
[b2]
[d3]
[g4]
[c5]
[h6]
[f7]
[a8]
true
```

```
true ;
[b1]
[d2]
[f3]
[h4]
[c5]
[a6]
[g7]
[e8]
true ;
[e1]
[g2]
[b3]
[f4]
[c5]
[a6]
[h7]
[d8]
true ;
[e1]
[g2]
[b3]
[f4]
[c5]
[a6]
[d7]
[h8]
true ;
[h1]
[b2]
[e3]
[c4]
[a5]
[g6]
[d7]
[f8]
true ;
[b1]
[g2]
[c3]
[f4]
[h5]
[e6]
[a7]
[d8]
true ;
[g1]
[b2]
[f3]
[c4]
[a5]
[d6]
[h7]
[e8]
true
```

```
?- print_board.
 _ _ _ _ _ _ _ _
1| | | | |Q| | | |
 _ _ _ _ _ _ _ _
2| |Q| | | | | | |
 _ _ _ _ _ _ _ _
3| | | | | |Q| | |
 _ _ _ _ _ _ _ _
4|Q| | | | | | | |
 _ _ _ _ _ _ _ _
5| | | | | | |Q| |
 _ _ _ _ _ _ _ _
6| | | |Q| | | | |
 _ _ _ _ _ _ _ _
7| | | | | | | |Q|
 _ _ _ _ _ _ _ _
8| | |Q| | | | | |
   h g f e d c b a

true ;
 _ _ _ _ _ _ _ _
1| | | | | |Q| | |
 _ _ _ _ _ _ _ _
2| | |Q| | | | | |
 _ _ _ _ _ _ _ _
3| | | | |Q| | | |
 _ _ _ _ _ _ _ _
4| | | | | | |Q| |
 _ _ _ _ _ _ _ _
5|Q| | | | | | | |
 _ _ _ _ _ _ _ _
6| | | |Q| | | | |
 _ _ _ _ _ _ _ _
7| |Q| | | | | | |
 _ _ _ _ _ _ _ _
8| | | | | | | |Q|
 _ _ _ _ _ _ _ _
   h g f e d c b a

true ;
 _ _ _ _ _ _ _ _
1| | | | | |Q| | |
 _ _ _ _ _ _ _ _
2| | | |Q| | | | |
 _ _ _ _ _ _ _ _
3| | | | | | |Q| |
 _ _ _ _ _ _ _ _
4|Q| | | | | | | |
 _ _ _ _ _ _ _ _
5| | |Q| | | | | |
 _ _ _ _ _ _ _ _
6| | | | |Q| | | |
 _ _ _ _ _ _ _ _
7| |Q| | | | | | |
 _ _ _ _ _ _ _ _
8| | | | | | | |Q|
 _ _ _ _ _ _ _ _
   h g f e d c b a
```

```
true ;
 _ _ _ _ _ _ _ _
1| | | | | | | |Q|
 _ _ _ _ _ _ _ _
2| |Q| | | | | | |
 _ _ _ _ _ _ _ _
3| | | | |Q| | | |
 _ _ _ _ _ _ _ _
4| | |Q| | | | | |
 _ _ _ _ _ _ _ _
5|Q| | | | | | | |
 _ _ _ _ _ _ _ _
6| | | | | | |Q| |
 _ _ _ _ _ _ _ _
7| | | |Q| | | | |
 _ _ _ _ _ _ _ _
8| | | | | |Q| | |
 _ _ _ _ _ _ _ _
   h g f e d c b a

true ;
 _ _ _ _ _ _ _ _
1| | | | | | | |Q|
 _ _ _ _ _ _ _ _
2| |Q| | | | | | |
 _ _ _ _ _ _ _ _
3| | | | |Q| | | |
 _ _ _ _ _ _ _ _
4|Q| | | | | | | |
 _ _ _ _ _ _ _ _
5| | | | | | |Q| |
 _ _ _ _ _ _ _ _
6| | | | |Q| | | |
 _ _ _ _ _ _ _ _
7| | |Q| | | | | |
 _ _ _ _ _ _ _ _
8| | | | | |Q| | |
 _ _ _ _ _ _ _ _
   h g f e d c b a

true ;
 _ _ _ _ _ _ _ _
1| | |Q| | | | | |
 _ _ _ _ _ _ _ _
2| | | | | | | |Q|
 _ _ _ _ _ _ _ _
3| | | |Q| | | | |
 _ _ _ _ _ _ _ _
4| | | | | | |Q| |
 _ _ _ _ _ _ _ _
5|Q| | | | | | | |
 _ _ _ _ _ _ _ _
6| | | | | |Q| | |
 _ _ _ _ _ _ _ _
7| |Q| | | | | | |
 _ _ _ _ _ _ _ _
8| | | | |Q| | | |
 _ _ _ _ _ _ _ _
   h g f e d c b a
```

Practical- 8

Aim: Universal and Existential Quantifier Variables in PROLOG.

Code:

File Edit View

```
member(X, [X | Xs]).
member(X, [Y | Xs]) :- member(X, Xs).
```

is understood as the set of logical axioms

∀X,Xs (member(X, [X | Xs]))

∀X,Y,Xs (member(X, Xs) => member(X, [Y | Xs])).

Output:

```
% C:/Users/Sonal Mishra/OneDrive/Desktop/labo.pl compiled 0.00 sec, 2 clauses
?- member(A,[1,2,3]), member(A,[B,3,4]).
A = B, B = 1 ;
A = B, B = 2 ;
A = B, B = 3 ;
A = 3 ;
false.

?- member(A,[4,5,6]), member(A,[B,5,6]).
A = B, B = 4 ;
A = B, B = 5 ;
A = 5 ;
A = B, B = 6 ;
A = 6 ;
false.
```

Practical- 9

Aim: Knowledge Base and Rule Base Creation for a specific domain in PROLOG.

Code:

```
male(homer).
male(bart).
male(abe).

female(marge).
female(lisa).
female(maggie).
female(mona).

parent(homer, bart).
parent(homer, lisa).
parent(homer, maggie).
parent(marge, bart).
parent(marge, lisa).
parent(marge, maggie).
parent(abe, homer).
parent(mona, homer).
mother(X, Y) :- parent(X, Y), female(X).
father(X, Y) :- parent(X, Y), male(X).
son(X, Y) :- parent(Y, X), male(X).
daughter(X, Y) :- parent(Y, X), female(X).

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

Output:

```
?- parent(homer,X).
X = bart ;
X = lisa ;
X = maggie.

?- grandparent(X,maggie).
X = abe ;
X = mona.

?- mother(marge,Y).
Y = bart ;
Y = lisa ;
Y = maggie.

?- son(bart,C).
C = homer ;
C = marge.

?- father(homer,C).
C = bart ;
C = lisa ;
C = maggie.

?- daughter(maggie,Z).
Z = homer ;
Z = marge.

?-
```

Practical- 10

Aim: Implementation of Resolution process in PROLOG.

Code:

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

parent(kim, holly).
parent(margaret, kim).
parent(margaret, kent).
parent(esther, margaret).
parent(herbert, margaret).
parent(herbert, jean).

grandparent(GP, GC) :- parent(P, GC), parent(GP, P).

greatGrandParent(GGP, GGC) :- parent(P, GGC), grandparent(GGP, P).

sibling(X, Y) :- parent(P, X), parent(P, Y), not(X=Y).
```

Output:

```
?-
|    not(parent(margaret, holly)).
true.

?- parent(X,margaret).
X = esther ;
X = herbert.

?- siblings(esther,herbert).
Correct to: "sibling(esther,herbert)"?
Please answer 'y' or 'n'? yes
false.

?- not(parent(margaret, holly)).
true.

?- sibling(esther,herbert).
false.

?- ancestor(X,margaret).
X = esther ;
X = herbert.

?-
```

Practical- 11

Aim: Implementation of an Expert System for a particular domain.

Code:

```
domains
disease,indication,name = symbol
predicates
hypothesis(name,disease)
symptom(name,indication)
clauses
symptom(amit,fever).
symptom(amit,rash).
symptom(amit,headache).
symptom(amit,runn_nose).
symptom(kaushal,chills).
symptom(kaushal,fever).
symptom(kaushal,hedache).
symptom(dipen,runny_nose).
symptom(dipen,rash).
symptom(dipen,flu).
hypothesis(patient,measels):-
symptom(patient,fever),
symptom(patient,cough),
symptom(patient,conjunctivitis),
symptom(patient,rash).
hypothesis(patient,german_measles) :-
symptom(patient,fever),
symptom(patient,headache),
symptom(patient,runny_nose),
symptom(patient,rash).
```

```
hypothesis(patient,flu) :-
symptom(patient,fever),
symptom(patient,headache),
symptom(patient,body_ache),
symptom(patient,chills).
hypothesis(patient,common_cold) :-
symptom(patient,headache),
symptom(patient,sneezing),
symptom(patient,sore_throat),
symptom(patient,chills),
symptom(patient,runny_nose).
hypothesis(patient,mumps) :-
symptom(patient,fever),
symptom(patient,swollen_glands).
hypothesis(patient,chicken_pox) :-
symptom(patient,fever),
symptom(patient,rash),
symptom(patient,body_ache),
symptom(patient,chills).
```

Output:

```
?-
|    symptom(kaushal,X).
X = chills ;
X = fever ;
X = hedache.

?- symptom(dipen,S).
S = runny_nose ;
S = rash ;
S = flu.

?- hypothesis(patient,X).
false.

?-  hypothesis(patient,mumps).
false.

?-
|    symptom(amit,X).
X = rash ;
X = headache ;
X = runn_nose.

?- symptom(amit,fever).
false.

?- symptom(amit,headache).
true .

?- symptom(dipen,X).
X = runny_nose ;
X = rash ;
X = flu.

?- symptom(kaushal,Z).
Z = chills ;
Z = fever ;
Z = hedache.
```

```
?- symptom(X,chills).
X = kaushal.

?- symptom(kaushal,flu).
false.

?- symptom(Z,rash).
Z = amit ;
Z = dipen.

?- symptom(X,runny_nose).
X = dipen.

?- symptom(X,runny_nose).
X = dipen.

?- symptom(S,sneezing).
false.

?- symptom(amit,cough).
false.

?-
```