PRACTICAL: 1

AIM: Write a PROLOG program that list four address in a label form; each address should list a name, one-line address, city, state &ZIP code.

CODE:

```
Domains
  Name,Society,City,State,ZipCode = string

Predicates
  getaddress(Name,Society,City,State,ZipCode)

Clauses
  soc(bhumit,gayatrinagar).
  soc(john,patelpark).
  soc(peter,goghacircle).

  city(bhumit,bhavnagar).
  city(john,bhavnagar).
  city(peter,surat).

  state(bhumit,gujrat).
  state(john,gujrat).
  state(peter,gujrat).

  zip(bhumit,364001).
  zip(john,364001).
  zip(peter,362002).

  getaddress(Name,Soci,Mycity,Sta,Code):-

  soc(Name,Soci),
  city(Name,Mycity),
  state(Name,Sta),
  zip(Name,Code).
```

OUTPUT:

```
% e:/pract-1.pl compiled 0.00 sec, 13 clauses
?- getaddress(bhumit,Society,City,State,ZipCode).
Society = gayatrinagar,
City = bhavnagar,
State = gujrat,
ZipCode = 364001.

?-
```

PRACTICAL: 2

AIM: WAP to Create Database for Hobbies of Different Person.

CODE:

```
Domains
    person = symbol
    hobby = symbol
Predicates
    likes (person,hobby)
Clauses
    likes(bhumit,cricket).
    likes(shyam,volleyball).
    likes(ram,basketball).
```

OUTPUT:

```
% e:/pract-2.pl compiled 0.00 sec, 3 clauses
?- likes(bhumit,cricket).
true.

?- likes(bhumit,volleyball).
false.

?-
```

PRACTICAL: 3

AIM: Write a Turbo PROLOG program for diagnosis the childhood diseases.

CODE:

```
Domains
    disease,indication,name = symbol
Predicates
    hypothesis(name,disease)
    symptom(name,indication)
 Clauses
    symptom(charlie,fever).
    symptom(charlie,headache).
    symptom(charlie,runnynose).
    symptom(charlie,rash).

    hypothesis(patient,measles):-
            symptom(Patient,fever),
            symptom(Patient,cough),
            symptom(Patient,conjunctive),
            symptom(Patient,runnynose),
            symptom(Patient,rash).

    hypothesis(Patient,germanmeasles):-
            symptom(Patient,fever),
            symptom(Patient,headache),
            symptom(Patient,runnynose),
            symptom(Patient,rash).

    hypothesis(Patient,flu):-
            symptom(Patient,fever),
            symptom(Patient,headache),
            symptom(Patient,bodyache),
            symptom(Patient,chills),
            symptom(Patient,sorethrought),
            symptom(Patient,cough),
            symptom(Patient,conjunctive),
            symptom(Patient,conjunctive),
            symptom(Patient,runnynose).

    hypothesis(Patient,commoncold):-
            symptom(Patient,headache),
            symptom(Patient,runnynose),
            symptom(Patient,snuzing),
```

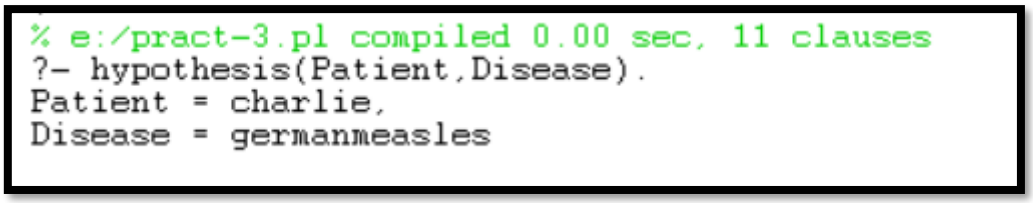
```
        symptom(Patient,chills),
        symptom(Patient,sorethrought).

hypothesis(Patient,mumps):-
        symptom(Patient,fever),
        symptom(Patient,swallenglands).

hypothesis(Patient,chikenpox):-
        symptom(Patient,fever),
        symptom(Patient,rash),
        symptom(Patient,bodyache).

hypothesis(Patient,whooping-cough):-
        symptom(Patient,runnynose),
        symptom(Patient,snuzing),
        symptom(Patient,cough).
```

OUTPUT:



PRACTICAL: 4

AIM: Write a Turbo PROLOG program for Family Relationship.

CODE:

```
Domains
    Person = symbol

Predicates
    male(person)
    female(person)
    parent(person,person)
    father(person,person)
    mother(person,person)
    sister(person,person)
    brother(person,person)
    son(person,person)
    daughter(person,person)
    aunt(person,person)
    uncle(person,person)
    child(person,person)
    wife_of(person,person)
    husband_of(person,person)
    grand_father(person,person)
    grand_mother(person,person)
    cousin(person,person)
    nephew(person,person)

Clauses
    father("Motilal","Jawaharlal").
    father("Motilal","Vijayalakshmi").
    father("Motilal","Krishna").
    father("Jawaharlal","Indira").
    father("Ranjit","Tara").
    father("Ranjit","Lekha").
    father("Ranjit","Rita").
    father("Feroz","Sanjay").
    father("Feroz","Rajiv").
    father("Sanjay","Varun").
    father("Rajiv","Rahul").
    father("Rajiv","Priyanka").

    wife_of("Swaruprani","Motilal").
    wife_of("Kamla","Jawaharlal").
```

```
wife_of("Vijayalakshmi","Ranjit").
wife_of("Indira","Feroz").
wife_of("Maneka","Sanjay").
wife_of("Sonia","Rajiv").

female("Krishna").
female("Priyanka").
female("Lekha").
female("Tara").
female("Rita").

female(X) :-
wife_of(X,_).
male("Varun").
male("Rahul").

male(X) :-
husband_of(X,_).

husband_of(X,Y) :-
wife_of(Y,X).

mother(X,Y):-
wife_of(X,Z),
father(Z,Y).

parent(X,Y):-
father(X,Y);
mother(X,Y).

child(X,Y):-
parent(Y,X).

son(X,Y):-
child(X,Y),
male(X).

daughter(X,Y):-
child(X,Y),
female(X).

brother(X,Y):-
father(Z,X),
father(Z,Y),
```

```
male(X),
not(X=Y).

sister(X,Y):-
father(Z,X),
father(Z,Y),
female(X),
not(X=Y).

uncle(X,Y):-
parent(Z,Y),
brother(X,Z);
parent(Z,Y),
sister(S,Z),
husband_of(X,S).

aunt(X,Y):-
sister(X,Z),
parent(Z,Y).

aunt(X,Y):-
wife_of(X,Z),
uncle(Z,Y).

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

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

grand_father(X,Y):-
parent(X,Z),
parent(Z,Y),
male(X).

grand_mother(X,Y):-
parent(X,Z),
parent(Z,Y),
female(X).

cousin(X,Y):-
parent(Z,X),
parent(W,Y),
```

```
brother(Z,W);
parent(Z,X);
parent(W,Y),
sister(Z,W).

nephew(X,Y):-
male(X),
uncle(Y,X);
male(X),
aunt(Y,X).

niece(X,Y):-
female(X),
uncle(Y,X);
female(X).
```

OUTPUT:

```
% e:/pract-4.pl compiled 0.00 sec, 87 clauses
?- father("Rajiv",Child).
Child = "Rahul" ;

?- father(Father,"Tara").
Father = "Ranjit" ;

?- wife_of(Wife,"Rajiv")
|    .
Wife = "Sonia"
```

PRACTICAL: 5

AIM: Write a prolog program to give an opportunity to user to re-enter the password three (03) times, on entering wrong password.

CODE:

```
Domains
    name, password = symbol

Predicates
    getinput,
    logon,
    user(name,password)

Clauses
    logon :- getinput,
          write('You are logged on.'),nl.
    logon :- repeat,
          write('Sorry, you are not permitted.'),nl,
          write('Try again.'),nl,
          getinput,
          write('You are now logged on.').
    getinput :- write('please enter your name : '),
          read(Name),nl,
          write('please enter password : '),
          read(Password),nl,
          user(Name, Password).

    user(bhumit,123).
```

OUTPUT:

```
% e:/pract-5.pl compiled 0.00 sec, 4 clauses
?- logon.
please enter your name : bhumit.

please enter password : |: 456.

Sorry, you are not permitted.
Try again.
please enter your name : |: smith.

please enter password : |: 123.

Sorry, you are not permitted.
Try again.
please enter your name : |: bhumit.

please enter password : |: 123.

You are now logged on.
true .
```

PRACTICAL: 6

AIM: Write a Turbo PROLOG program to implement Tower of Hanoi problem.

CODE:

```
Domains
    POLE = symbol

Predicates
    move(INTEGER, POLE, POLE, POLE)

Clauses
    move(1,X,Y,_):-write('Move disk from '),write(X),write(' to '),write(Y),nl.
    move(N,X,Y,Z):-N>1,M is N-1,
            move(M,X,Z,Y),
            move(1,X,Y,_),
            move(M,Z,Y,X).
```

OUTPUT:

```
% e:/pract-6.pl compiled 0.00 sec, 2 clauses
?- move(3,a,b,c).
Move disk from a to b
Move disk from a to c
Move disk from b to c
Move disk from a to b
Move disk from c to a
Move disk from c to b
Move disk from a to b
true
```

PRACTICAL: 7

AIM: Write a Turbo PROLOG program to solve Water-Jug Problem.

CODE:

```
Domains

Predicates
    jug(INTRGER, INTRGER)

Clauses
    jug(2, _).

    jug(0,2):-
        write('(0,2)'),nl,
        write('(2,0)'),nl.

    jug(4,0) :-
        write('(4,0)'),nl,
        jug(0,0).

    jug(4,3) :-
        write('(4,3)'),nl,
        jug(0,0).

    jug(3,0) :-
        write('(3,0)'),nl,
        jug(3,3).

    jug(X,0) :-
        write('('),write(X),write(',0'),nl,
        jug(0,3).

    jug(0,3) :-
        write('(0,3)'),nl,
        jug(3,0).
    jug(0,X) :-
        write('(0,'),write(X),write(')'),nl,
        jug(0,0).

    jug(3,3) :-
        write('(3,3)'),nl,
        jug(4,2).
```

```
jug(4,2) :-
    write('(4,2)'),nl, write('2,0'), nl,
    jug(2,0).

jug(X, Y) :-
    X>4,fail,Y>3,fail.
```

OUTPUT:

```
% e:/pract-7.pl compiled 0.00 sec, 11 clauses
?- jug(1,1).
false.

?- jug(4,3).
(4,3)
(0,0
(0,3)
(3,0)
(3,3)
(4,2)
2,0
true
```

PRACTICAL: 8

AIM: Implement Depth First Search and Breadth algorithm in choice of your language.

1. **BFS:**

CODE:

```
Domains
Predicates
Clauses
    s(a, b).
    s(a, c).
    s(b, g).
    s(b, f).
    s(c, r).
    s(c, e).
    goal(f).
    solve( Start, Solution) :-
    breadthfirst( [ [Start] ], Solution).
    breadthfirst( [ [Node | Path] |_], [Node | Path] ) :-
    goal( Node).
    breadthfirst( [ [N | Path] | Paths], Solution) :-
    bagof([M,N|Path],
    ( s( N, M), \+ member( M, [N | Path] ) ), NewPaths),
       %conc( Paths, NewPaths, Pathsl), !,
    append(Paths, NewPaths, Pathsl), !,
    breadthfirst( Pathsl, Solution);
    breadthfirst( Paths, Solution).
```

OUTPUT:

2. DFS:

CODE:

```
Domains
Predicates
Clauses
    connected(1,7,1).
    connected(1,8,1).
    connected(1,3,1).
    connected(7,4,1).
    connected(7,20,1).
    connected(7,17,1).
    connected(8,6,1).
    connected(3,9,1).
    connected(3,12,1).
    connected(9,19,1).
    connected(4,42,1).
    connected(20,28,1).
    connected(17,10,1).

    connected2(X,Y,D) :- connected(X,Y,D).
    connected2(X,Y,D) :- connected(Y,X,D).

    next_node(Current, Next, Path) :-
       connected2(Current, Next, _),
    not(member(Next, Path)).
    depth_first(Goal, Goal, _, [Goal]).
    depth_first(Start, Goal, Visited, [Start|Path]) :-
    next_node(Start, Next_node, Visited),
    write(Visited), nl,
    depth_first(Next_node, Goal, [Next_node|Visited], Path).
```

OUTPUT:

```
% e:/pract-8(b).pl compiled 0.00 sec, 18 clauses
?- depth_first(1,28,[1],P).
[1]
[7,1]
[4,7,1]
[7,1]
[20,7,1]
P = [1, 7, 20, 28]
```