PRACTICAL MANUAL

OF

ARTIFICIAL INTELLIGENCE

"AI is the part of computer science concerned with designing intelligent computer systems, that is, computer systems that exhibit the characteristics we associate with intelligence in human behaviour - understanding language, learning, reasoning and solving problems"

GOVERTMENT OF ENGINNERING COLLEGE
DEPARTMENT OF C.E/I.T
VII SEMESTER
MODASA

INDEX

15		Write a Turbo PROLOG program for finding the average salary of an employee and for adding and deleting employees from the database.		
16		Write a Turbo PROLOG program to solve Water-Jug Problem.		
17		Write a Turbo PROLOG program to demonstrate the effective use of Cut and Fail.		
18		Write a Turbo PROLOG program for Traveling Salesman Problem.		
19		Write a Turbo PROLOG program for Monkey Banana Problem.		
20		Write a Turbo PROLOG program N-QUEEN problem.		

Practical - 1

1. WAP to Create Database for Hobbies of Different Person.

Domains

Person, hobbies=symbol

Predicates

Likes (person, hobbies)

Clauses

Likes (vijay, chess).
Likes (ajay, cricket).

2) Output:

Goal:

Likes (vijay, chess)
Yes
Solution

Likes (ajay, chess)
No
Solution

:

Practical - 2

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

```
domains
        person = address(name, street, city, state, zip)
        name, street, city, state, zip = string
predicates
        readaddress(person)
        go
clauses
        go:-
                readaddress(Address), nl,
                write(Address), nl, nl,
                write("Accept (y/n) ? "),
                readchar(Reply), Reply = 'y', !.

        go:-
                nl, write("Please reenter"), nl,
                go.

        readaddress(address(Name, Street, City, State, Zip)) :-
                write("Name : "), readln(Name),
                write("Street : "), readln(Street),
                write("City : "), readln(City),
                write("State : "), readln(State),
                write("Zip : "), readln(Zip).
```

2) Output:

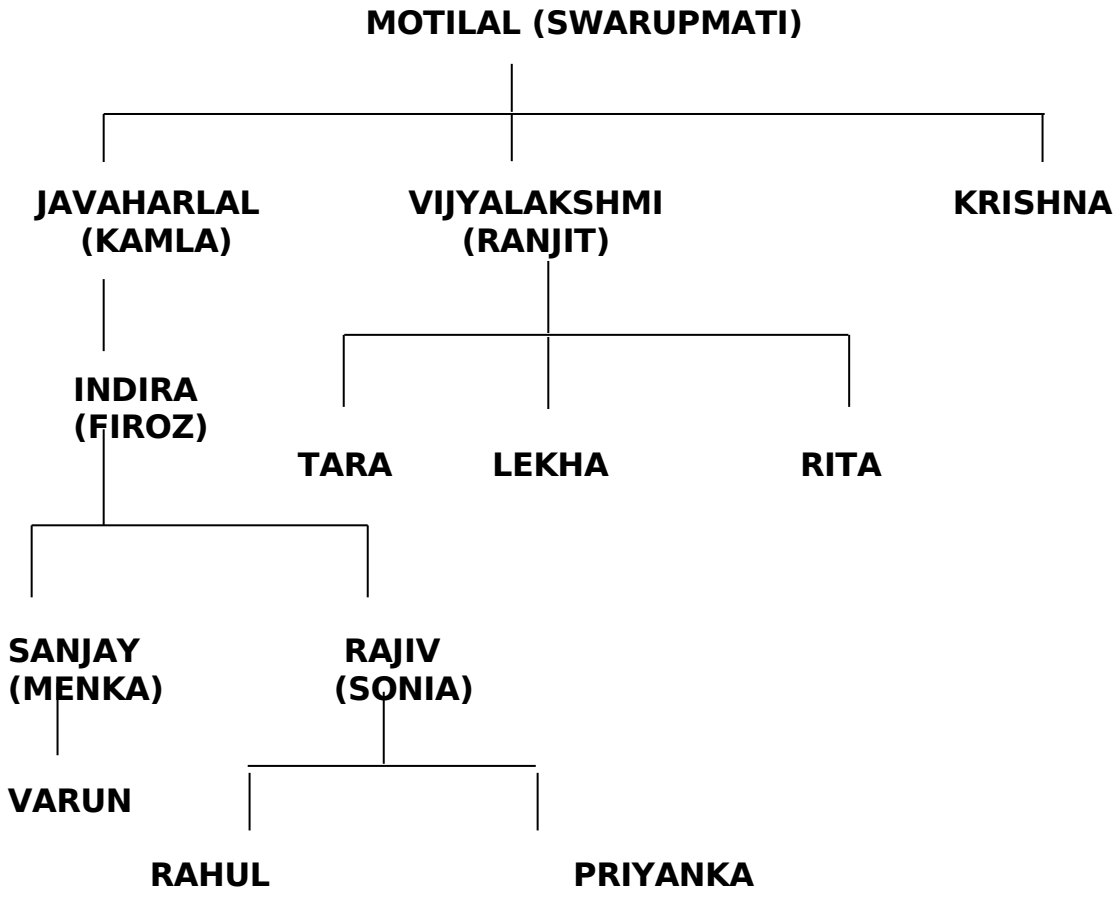
Goal: go
Name: bhavik
Street: Naranpura
City: Ahmedabad
State: Gujarat
Zip: 380015

address("Bhavik ,"Naranpura","Ahmedabad","Gujarat","380015")

Accept (y/n) ? Yes
Goal:

Practical - 3

Write a Turbo PROLOG program for Family Relationship.




```
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)
```

```
        ancestor(person,person)
        wife_of(person,person)
        husband_of(person,person)


        grand_father(person,person)
        grand_mother(person,person)
        cousin(person,person)
        nephew(person,person)
        niece(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),
        aunt(Y,X).
```

1) Output:

Goal: father("Rajiv",Child)
Child=Rahul
Child=Priyanka
2 Solutions

Goal: father(Father,"Varun")
Father=Sanjay
1 Solution

Practical- 4

Write a Turbo PROLOG program for diagnosis the childhood diseases.

```
domains
        disease,indication,name=symbol
predicates
        hypothesis(name,disease)
        symptom(name,indication)
clauses
        symptom(charlie,fever).
        symptom(charlie,rash).
        symptom(charlie,headache).
        symptom(charlie,runny_nose).

        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,cough).
```

4) Output:

Goal: hypothesis(Patient,Disease)
Patient=charlie, Disease=german_measles
1 Solution

Practical - 5

Write a Turbo PROLOG program to calculate the roots of quadratic equation Consider all possibilities real, equal, imaginary.

```
domains
predicates
        root(real,real,real,real)
        run
clauses
        run:-
        write("Enter the value of A :" ),
        readreal(A),
        write("Enter the value of B :" ),
        readreal(B),
        write("Enter the value of C :" ),
        readreal(C),
        D = (B*B)-(4*A*C),
        root(A,B,C,D).

        root(A,B,C,D):-
        A=0.0,
        write("Only one root exists."),
        ANS = (-C/B),
        write(ANS)
        ;

        D>=0,
        ANS = (-B - sqrt(D)) / (2*A),
        ANS1 = (-B + sqrt(D)) / (2*A),
        write("First root is : "),
        write(ANS),nl,
        write("Second root is : "),
        write(ANS1)
        ;

        REAL= (-B) / (2*A),
        IMG = sqrt(-D) / (2*A),
        write("Real root is : "),
        write(REAL),nl,
        write("Imaginary root is : "),
        write(IMG).
```

5) Output:

Goal: run
Enter the value of A :1
Enter the value of B :-2

Enter the value of C :1
First root is : 1
Second root is : 1
Yes

Practical - 6

Write a Turbo PROLOG program based on list 1: -

A) To find the length of a list.

```
domains
        symbolist = symbol *
        integerlist = integer *
predicates
        Length ( symbolist , integer )
        Length ( integerlist , integer )
clauses
        Length ( [ ], 0 ).
        Length ( [_| Tail ],N):-
        Length ( Tail , N1 ),
        N = 1 + N1.
```

B) To find whether given element is a member of a list.

```
domains
        namelist=symbol*
predicates
        member ( symbol , namelist )
        club_member(symbol)
clauses
        member ( X, [X|_ ] ).
        member ( X, [_| Tail ] ):-
                member ( X, Tail ).
        club_member(Name):-
                member(Name,[bindu,swati,rita]).
```

6) Output:

A)

Goal: Length([a,b,c,d],M)
M=4
1 Solution

B)

Goal: club_member(bindu)
Yes
Goal: club_member(aa)
No

Practical - 7

Write a Turbo PROLOG program based on list: -

C) To Append the list.

```
domains
        namelist=symbol*
predicates
        append ( namelist , namelist,namelist )
clauses
        append ( [ ], ListB, ListB ).
        append ( [ X | List1 ], List2 ,[ X | List3 ] ):-
        append (List1,List2,List3 ).
```

D) To Reverse the list.

```
domains
        namelist=symbol*
predicates
        reverse_list(namelist,namelist)
        reverse(namelist,namelist,namelist)
clauses
        reverse_list(Inputlist ,Outputlist):-
        reverse(Inputlist,[],Outputlist).
        reverse([],Inputlist,Inputlist).
        reverse ( [Head | Tail ],List1, List2 ):-
        reverse ( Tail,[Head | List1], List2 ).
```

7) Output:

C)

```
Goal: append([g,h],[t,y,u],Append_List)
Append_List=["g","h","t","y","u"]
1 Solution
```

D)

```
Goal: reverse_list([f,j,l],L)
L=["l","j","f"]
1 Solution
```

Practical - 8

Write a Turbo PROLOG program based on list 3: -

E) To find the last element of a list.

```
domains
        namelist=symbol*
predicates
        last(namelist,symbol)

clauses
        last([Head],X):-
                X=Head.
        last([_|Tail],X):-
                last(Tail,X).
```

F) To delete the first occurrence of an element from a list.

```
domains
        symbollist=symbol*
predicates
        delete(symbol,symbollist,symbollist)
clauses
        delete(_,[],[]).
        delete(X,[X|L],L):-!.
        delete(X,[Y|L],[Y|M]):-
        delete(X,L,M).
```

8) Output:

E)

```
Goal: last([f,j,k],Last_Element)
Last_Element=k
1 Solution
```

F)

Goal: delete(r,[r,n,r,m],F)

```
F=["n","r","m"]
1 Solution

Goal: delete(r,[n,f,m],F)
F=["n","f","m"]
1 Solution
```

Practical – 9

Write a Turbo PROLOG program Checking for Password.

A) Give an opportunity to user to re-enter the password 'n' no. Of times, on entering wrong password.

```
domains
        name,password = symbol
predicates
        getinput(name,password)
        logon
        user(name,password)
        repeat
clauses
        repeat.
        repeat:-
                repeat.
        logon:-
                clearwindow,
                getinput(_,_),
                write("You are now logged on."),nl.

        logon:-
                repeat,
                write("Sorry,you are not permitted access."),nl,
                write("Please try again."),nl,
                getinput(_,_),
                write("You are now logged on.").


        getinput(Name,Password):-
                write("Please enter your name : "),
                readln(Name),nl,
                write("Please enter password : "),
                readln(Password),nl,
                user(Name,Password).

        user(vinod,patil).
        user(himmat,solanki).
```

Output:

9)

A)

```
Please enter your name: vinod
Please enter password: patel
Sorry, you are not permitted access.
Please try again.
Please enter your name: himmat
Please enter password: solanki
You are now logged on.
Yes
```

Practical –10

Write a Turbo PROLOG program Checking for Password.

B) Give an opportunity to user to re-enter the password three (03) Times, on entering wrong password.

```
Domains
        name, password = symbol
predicates
        getinput(name,password)
        Logon (integer)
        user (name, password)
        go
clauses
        go:-
                logon(3),
                write("You are now logged on."),nl.

        Logon(0):-!,
                write("Sorry, you are not permitted access."),
                fail.

        Logon(_):-
                getinput(Name,Password),
                user(Name,Password).

        Logon(X):-
                write("Illegal entry."),nl,
                XX = X – 1,
                logon(XX).

        Getinput(Name,Password):-
                write("Please enter your name : "),
                readln(Name),
                write("Please enter password : "),
                readln(Password).

        User(harsh,patel).
```

User(jacky,singh).

10)
B)

Goal: go
Please enter your name : aa
Please enter password : bb
Illegal entry.
Please enter your name : cc
Please enter password : ss
Illegal entry.
Please enter your name : dd
Please enter password : gg
Illegal entry.
Sorry,you are not permitted access.
No

Goal: go
Please enter your name :harsh
Please enter password : patel
You are now logged on.
Yes

Practical - 11

Write a Turbo PROLOG program for Arithmetic Operations.

A) To add the member of a given list.

```
domains
        integerlist = integer*
        reallist = real*
predicates
        sum(integerlist,integer)
        sum(reallist,real)
clauses
        sum([],0).
        sum([Head | Tail],Res):-
                sum(Tail,SumTail),
                Res=Head+Sumtail.
```

B) To check if a given year is a Leap Year or not.

```
domains
predicates
        run
        leap(integer)
clauses
        run:-
                write("Type in the year : "),
                readint(Y),
                leap(Y),
                write("This year is a leap year."),nl.
        run:-
                write("This is not a leap year. "),nl.
        leap(Y):-
                X=Y mod 100,
                X=0, ! ,
                X=Y mod 400,
                X=0 , ! .
```

```
leap(Y):-
        X= Y mod 4,
        X=0.
```

C) To find the Greatest Common Divisor.

```
domains
predicates
        gcd (integer,integer,integer)
clauses
        gcd(M,0,M).
        gcd(M,N,Result):-
                N>0,
                Rem=M mod N,
                gcd(N,Rem,Result).
```

11) Output:

A)

Goal: sum([2,5,6,7],Result)
Result=20
1 Solution

Goal: sum([],Result)
Result=0
1 Solution

B)

Goal: run
Type in the year : 2004
This year is a leap year.
Yes

Goal: run
Type in the year : 2001
This is not a leap year.
Yes

C)

Goal: gcd(5,3,GCD)
GCD=1
1 Solution

Goal: gcd(4,12,GCD)
GCD=4
1 Solution

Practical - 12

Write a Turbo PROLOG program for Arithmetic Operations.

D) To find the Least Common Divisor.

```
domains
predicates
        lcd(integer,integer,integer)
        gcd(integer,integer,integer)
clauses
        gcd(M,0,M).
        gcd(M,N,Result):-
                N>0,
                Rem = M mod N,
                gcd(N,Rem,Result).
        lcd(M,N,Result):-
                gcd(M,N,Res1),
                Result = M * N / Res1.
```

E) To find the factorial of a given number.

```
domains
predicates
        factorial(integer,integer)
clauses
        factorial(0,1).
        factorial(N,F) :-
          N>0,
          N1 = N-1,
          factorial(N1,F1),
          F = N * F1.
```

F) To generate the Fibonacci series of a given number.

```
domains
      A,B,N=integer
predicates
      go
      fibo(integer, integer,integer, integer)
clauses
      go:-
            write("Enter term no : "),
            readint(N),
            fibo(1,1,0,N).

      fibo(_,_,_,0).
      fibo(A,B,C,N):-
            AA = B,
            BB = C,
            CC=AA+BB,
            write(CC," "),
            NN = N - 1,
            fibo(AA,BB,CC,NN).
            fibo(AA,BB,CC,NN).
```

G) To convert an integer number into a string of equivalents binary

```
predicates
                decimaltobinary(integer)
clauses
                decimaltobinary(0).
                decimaltobinary(D):-
                 D>0,
                 Digit=D mod 2,
                 I1=D div 2,
                 decimaltobinary(I1),
                  write(Digit).
```

12) Output:

D)

Goal: lcd(4,12,LCD)
LCD=12
1 Solution

E)

Goal: factorial(5,K)
K=120
1 Solution
Goal: factorial(0,K)
K=1
1 Solution

F)

Enter term no : 8
1 1 2 3 5 8 13 21
Yes

G)

Goal: decimaltobinary(12)
1100Yes
Goal:

Practical – 13

Write a program to find Minimum from give Numbers.

```
domains
        R = real
predicates
        run
        mini(integer,integer,integer)
clauses
        run :-
                write("Enter the first number :"),
                readreal(M),
                write("Enter the second number :"),
                readreal(N),
                mini(M,N,_).

        mini(M,N,R):-
           M<N,
                R=M,
                write("Minimum number is : ",R),nl
                ;
                M>N,
                R=N,
                write("Minimum number is : ",R),nl
                ;
           M=N,
                R=M,
                write("Minimum number is : ",R),nl.
```

13) Output:

Goal: run
Enter the first number :10
Enter the second number :11
Minimum number is : 10
Yes

Goal: run
Enter the first number :7
Enter the second number :7
Minimum number is : 7
Yes

Practical – 14

Write a Turbo PROLOG program to implement Tower Of Hanoi Problem.

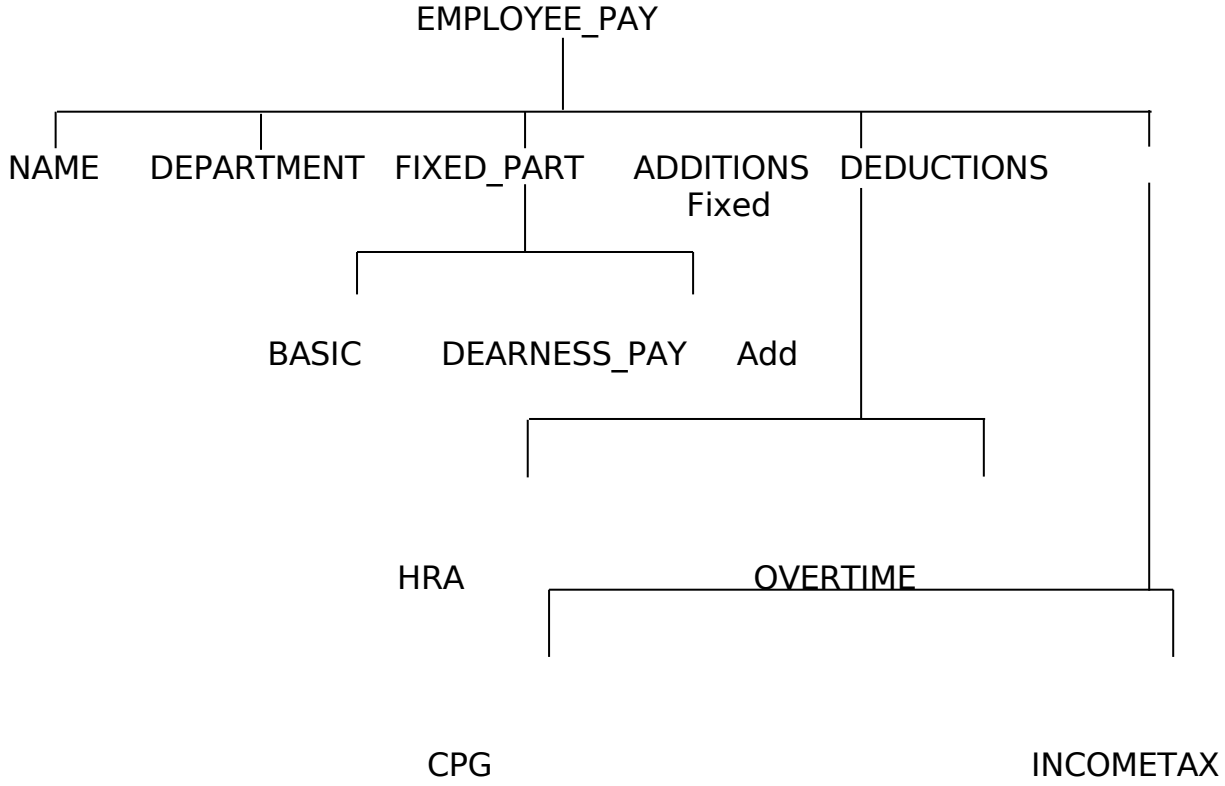
```
domains
    POLE = SYMBOL
predicates
    hanoi
    hanoi(INTEGER)
    move(INTEGER,POLE,POLE,POLE)
    inform(POLE,POLE)
clauses
    hanoi:-hanoi(5).
    hanoi(N) :-
        move(N, left, middle, right).
    move(1, A, _, C) :-
        inform(A, C), !.
    move(N, A, B, C) :-
        N1 = N-1,
        move(N1, A, C, B),
        inform(A, C), !,
        move(N1, B, A, C).
    inform(Loc1, Loc2) :-
    write("Move a disk from ", Loc1, " to ", Loc2),nl, !.
```

14) Output:

Goal: hanoi(3)
Move a disk from left to right
Move a disk from left to middle
Move a disk from right to middle
Move a disk from left to right
Move a disk from middle to left
Move a disk from middle to right
Move a disk from left to right
Yes

Practical - 15

Write a Turbo PROLOG program for finding the average salary of an employee and for adding and deleting employees from the database.




```
domains
        name,dept=symbol
        cpg,it,hr,ot,basic,dp=real
```

```
        ded=deduct(cpg,it)
        addition=add(hr,ot)
        fp=fixed(basic,dp)
        basiclist=basic*
database
        emp_pay(name,dept,fp,addition,ded)
predicates
        add_emp
        del_emp
        sum(basiclist,real)
        len(basiclist,integer)
        avg
        finish
        run(basiclist,real)
clauses
        emp_pay("paras","IT",fixed(24000,1080),add(8000,1155),deduct(4000,1000)).
        emp_pay("neeraj","FIN",fixed(25000,1080),add(5000,1155),deduct(4000,900)).
        emp_pay("hardik","HR",fixed(26000,1080),add(7000,1155),deduct(2000,1000)).
        emp_pay("vipul","RnD",fixed(28000,1080),add(4000,1155),deduct(1000,500)).
        emp_pay("rajen","IT",fixed(23000,1080),add(9000,1145),deduct(2500,200)).
        emp_pay("paresh","HR",fixed(26000,1080),add(8000,1055),deduct(1000,1000)).
 avg:-
         findall(Basic,emp_pay(_,_,fixed(Basic,_),_,_),L),
         run(L,Avg),
         write(Avg).
 run(L,A):-
         sum(L,Sum),
         len(L,N),
         A = Sum / N.
 add_emp:-
         write("Enter the name of employee :"),
         readln(Name),
         write("Enter the dept name:"),
         readln(Dept),
         write("Enter basic pay:"),
         readreal(Basic),
         write("Enter dearness pay:"),
         readreal(DP),
         write("Enter house rent:"),
         readreal(HR),
         write("Enter overtime:"),
         readreal(OT),
         write("Enter CPG deducted:"),
         readreal(CPG),
         write("Enter income tax:"),
         readreal(IT),
         assertz(emp_pay(Name,Dept,fixed(Basic,DP),add(HR,OT),deduct(CPG,IT))),
         write("Do u want to add another employee(Y/N)?\n"),
```

```
        readchar(Reply),
        Reply='y',
        add_emp;
        exit.
del_emp:-
        write("Enter the name of employee:"),
        readln(Name),
        retract(emp_pay(Name,_,_,_,_)).
finish:-
        save("c:\output.dat").
        sum([],0).
        sum([H|T],Sum):-
        sum(T,ST),
        Sum=H+ST,
        len([],0).
        len([_|T],N):-
        len(T,N1),
        N=N1.
```

15) Output:

```
Goal: add_emp
Enter the name of employee :rishabh
Enter the dept name:HR
Enter basic pay:20000
Enter dearness pay:2000
Enter house rent:2000
Enter overtime:1000
Enter CPG deducted:2000
Enter income tax:3000
Do u want to add another employee(Y/N)?
Goal: del_emp
Enter the name of employee:paras
Yes
Goal:
```

Practical – 16

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

```
domains
predicates
        jug(integer, integer)
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("(", X, ", 0)"), nl,
                jug(0, 3).
        jug(0, 3):-
                write("(0, 3)"), nl,
                jug(3, 0).
        jug(0, X):-
                write("(0, ", X, ")"), 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.
```

16) Output:

```
Goal: jug(4,3)
(4, 3)
(0, 0)
(0, 3)
(3, 0)
(3, 3)
(4, 2)
(2, 0)
Yes
Goal: jug(1,1)
No
```

Practical – 17

Write a Turbo PROLOG program to demonstrate the effective use of Cut and Fail.

```
domains
        state,city = string
predicates
        location(city,state)
        go
        checkstate(state)
clauses
        go:-
                writef(" %-15 %5 \n","CITY","STATE"),
                fail.
        go :-
                location(City,State),
                checkstate(State),
                writef(" %-15 %5 \n",City,State),
                fail.
        go.
        location("Gandhinagar","Gujarat").
        location("Surat","Gujarat").
        location("Bombay","Maharastra").
        checkstate("DC") :-
                        fail.
        checkstate(_).
```

17) Output:

Goal: go

CITY	STATE
Gandhinagar	Gujarat
Surat	Gujarat
Bombay	Maharastra

Yes

Practical – 18

Write a Turbo PROLOG program for Traveling Salesman Problem.

```
domains
        X,City1,City2=string
        Dist,Dist1,Dist2=integer
predicates
        route(string,string,integer)
        read(string,string,integer)
        find
clauses
        read("Surat","Baroda",150).
        read("Ahemedabad","Baroda",150).
        read("Valsad","Bombay",300).
        read("Bombay","Ahemedabad",500).
        read("Baroda","Surat",150).
        read("Surat","Valsad",100).
        read("Bombay","Pune",100).

        find:-
                write("Source center:"),
                readln(City1),
                write("Dest center:"),
                readln(City2),
                route(City1,City2,Dist).
        route(City1,City2,Dist):-
                read(City1,City2,Dist),
                writef("DISTANCE betn %10 and %10 is %2",City1,City2,Dist),nl.
```

```
route(City1,City2,Dist):-
        read(City1,X,Dist1),
        route(X,City2,Dist2),
        Dist=Dist1+Dist2,
        writef("The dist betn %10 and %10 is %2",City1,City2,Dist).
```

18) Output:

```
Goal: find
Source center:Surat
Dest center:Valsad
DISTANCE betn      Surat
 and     Valsad is 100
Yes

Goal: find
Source center:Bombay
Dest center:Pune
DISTANCE betn     Bombay
 and       Pune is 100
Yes
```

Practical–19

Write a Turbo PROLOG program for Monkey Banana Problem.

```
domains
predicates
        in_room(symbol)
        dexterous(symbol)
        tall(symbol)
        can_move(symbol,symbol,symbol)
        can_climb(symbol,symbol)
        can_reach(symbol,symbol)
        close(symbol,symbol)
        get_on(symbol,symbol)
        under(symbol,symbol)
clauses
        in_room(bananas).
        in_room(chair).
        in_room(monkey).
        dexterous(monkey).
        tall(chair).
        can_move(monkey,chair,bananas).
        can_climb(monkey,chair).
        can_reach(X,Y):-
                dexterous(X),
                close(X,Y).
        close(X,Z):-
                get_on(X,Y),
                under(Y,Z),
```

```
        tall(Y).
get_on(X,Y):-
        can_climb(X,Y).
under(Y,Z):-
        in_room(X),
        in_room(Y),
        in_room(Z),
        can_move(X,Y,Z).
```

19) Output:

```
Goal: can_reach(monkey,bananas)
Yes
Goal: can_reach(lion,bananas)
No
Goal: can_reach(X,Y)
X=monkey, Y=bananas
1 Solution
```

Practical –20

Write a Turbo PROLOG program N-QUEEN problem.

```
domains
        queen    = q(integer,integer)
        queens   = queen*
        freelist = integer*
        board    = board(queens,freelist,freelist,freelist,freelist)
predicates
        placeN(integer,board,board)
        place_a_queen(integer,board,board)
        nqueens(integer)
        makelist(integer,freelist)
        findandremove(integer,freelist,freelist)
        nextrow(integer,freelist,freelist)
clauses
        nqueens(N) :-
                makelist(N,L),
                Diagonal=N*2-1,
                makelist(Diagonal,LL),
                placeN(N,board([],L,L,LL,LL),Final),
                write(Final).
        placeN(_, board(D,[],[],D1,D2),board(D,[],[],D1,D2)):- !.
        placeN(N, Board1, Result) :-
                place_a_queen(N,Board1,Board2),
                placeN(N,Board2,Result).

        place_a_queen(N,board(Queens,Rows,Columns,Diag1,Diag2),
```

```
  board([q(R, C)|Queens],NewR,NewC,NewD1,NewD2)) :-
        nextrow(R,Rows,NewR),
        findandremove(C,Columns,NewC),
        D1 = N+C-R,
        findandremove(D1,Diag1,NewD1),
        D2 = R+C-1,
        findandremove(D2,Diag2,NewD2).

findandremove(X,[X|Rest],Rest).
findandremove(X,[Y|Rest],[Y|Tail]) :-
        findandremove(X, Rest,Tail).

makelist(1,[1]).
makelist(N,[N|Rest]) :-
        N1 = N-1,makelist(N1,Rest).

nextrow(Row, [Row|Rest], Rest).
```

20) Output:

```
Goal: nqueens(8)
board([q(1,5),q(2,7),q(3
,2),q(4,6),q(5,3),q(6,1)
,q(7,4),q(8,8)])Yes
Goal:
```