LAB MANUAL

Name: SHAH HETVI

Enrollment No.: 180850131016

Class: CSE sem. 7

Subject: Artificial Intelligence

Subject Code: 3170716

HJD INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH, KERA
COMPUTER ENGINEERING DEPARTMENT

HJD INSTITUTE OF TECHNICAL
EDUCATION AND RESEARCH, KERA

Approved by AICTE and affiliated to
Gujarat Technological University

CERTIFICATE

This is to certify that **Miss SHAH HETVI Enrollment No 180850131016** of programme **BE 7th SEM Computer Engineering** has satisfactorily completed **her** term work in **Artificial Intelligence(3170716)** for the term ending in 2021-2022.

Miss. Bhagyashree B Mirani	Mr. Vishal Bhimani
(Lab In-charge)	(Head Of Department)

INDEX

PRACTICAL 1: STUDY OF PROLOG.

Using Turbo
Prolog Topics:
a) Basics of Turbo Prolog
b) Intro to Prolog programming
c) Running a simple program

- Prolog is a logical programming language and stands for PROgramming in LOGic
- created around 1972
- Prelerred for Al programming and mainly used in such areas as:
  0 Theorem proving, expert systems, NLP, .
- Logical programming is the use of mathematical logic for computer programming.

To start Turbo Prolog. open a MSDOS window and type: N> prolog followed by a carriage return.

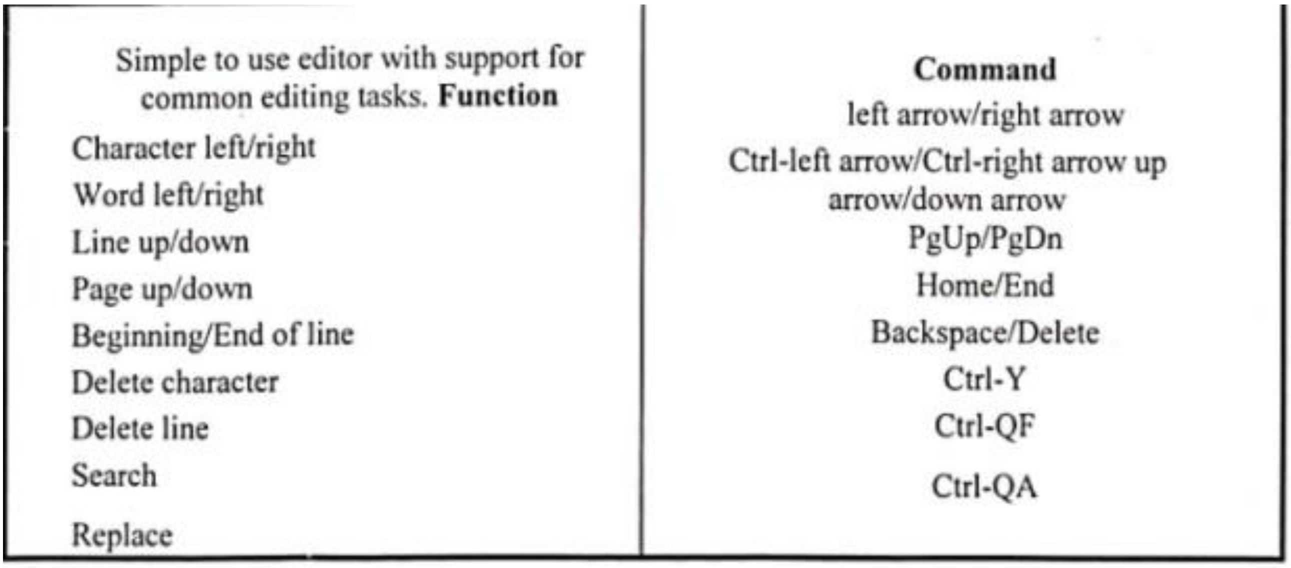
The GUI:

- Gui is composed of four panels and a main menu bar.
- The menu bar lists six options- Files, Edit, Run. Compile, Options, Setup.
- The four panels are Editor, Dialog. Message and Trace.

MENU

- **Files**-Enables the user to load programs from disk, create new programs, save modified program to disk, and quit the program.
- **Edit** - Moves user control to the Editor panel
- **Compile**-Provides the user with choices on how to save the compiled version of the program.
- **Options**-Provides the user with choices on the type of compilation to be used.
- **Setup**-Enables the user to change panel sizes, colors, and positions.

Editor



Simple to use editor with support for common editing tasks.

Function	**Command**
Character left/right	left arrow/right arrow
Word left/right	Ctrl-left arrow/Ctrl-right arrow
Line up/down	up arrow/down arrow
Page up/down	PgUp/PgDn
Beginning/End of line	Home/End
Delete character	Backspace/Delete
Delete line	Ctrl-Y
Search	Ctrl-QF
Replace	Ctrl-QA

Dialog

- When a P'rolog program is executing, output will be shown in the Dialog Panel

Message

- The Message Panel keeps the programmer up to date on processing activity.

Trace

- The Trace Panel is useful for finding problems in the programs you create.

Prolog Clauses

Any factual expression in Prolog is called a clause.
There are two types of factual expressions: facts and rules.

There are three categories of statements in Prolog:

- **Facts**: Those are true statements that form the basis tor the knowledge base.

- **Rules**: Similar to functions in procedural programming (C Java,)and has the form of if/then.

- **Queries**: Questions that are passed to the interpreter to access the knowledge base and start the program.

What is a Prolog program?

- Prolog is used for solving problems that involve objects and the relationships between objects.

- Program corsists of a database containing one or more facts and zero or more rules(next week).

- A fact is a relationship among a collection of objects. A fact is a one-line statement that ends with a full-stop.

```
parent (john, bart).
parent (barbara, bart).
male (john).
        dog(fido). >>Fido is a dog or It is true that fido is a dog
sister(mary, joe). >>Mary is Joe' s Sister.
play(mary, Joe, tennis). >> It is true that Mary and Joe play tennis.
```

- Relationships can have any number of objects.

- Choose names that are meaningful - because in Prolog names are arbitrary strings but people will have to associate meaning to them.

Facts..
Syntax rules:

1. The names of all relationships and objects must begin with a lower case letter. For example: likes, John, rachel.

2. The relationship is written first, and the objects are written separated by commas, and the objects are enclosed by a pair ot round brackets.

3. The character '.' must come at the end of each fact.

Terminology

1) The names of the objects that are enclosed within the round brackets in each fact are called arguments.

2) The name of the relationship, which comes just before the round brackets, is called the predicate.

3) The arguments of a predicate can either be names of objects (constants) or variables.

4) When defining relationships between objects using facts, attention should be paid to the order in which the objects are listed. While programming the order is arbitrary, however the programmer must decide on some order and be consistent.

5) Ex. likes(tom. anna). >> The relationship defined has a different meaning if the order of the objects is changed. Here the first object is understood to be the "liker". If we wanted to state that Anna also likes Tom then we would have to add to our database-likes(anna, tom).

6) Remember that we must determine how to interpret the names of objects and relationships.

Constants & Variables

Constants

- Constants are names that begin with lower case letters.
- Names of relationships are constants

Variables

- Variables take the place of constants in facts.
- Variables begin with upper case letters.

Turbo Prolog Program

A Turbo Prolog program consists of two or more sections. **Clauses**

Section

- The main body of the prolog program.
- Contains the clauses that define the program -facts and rules.

Predicates Section

- Predicates (relations) used in the clauses section are defined.
- Each relation used in the clauses of the clauses section must have a corresponding predicate definition in the predicates section. Except for the built in predicates of Turbo Prolog.

A predicate definition in the predicates section does not end with a period.

Predicate definitions contain different names than those that appear in the clauses section. Make Sure that the predicate definition contains the same number of names as the predicate does when it appears in the clauses Section.

A Turbo Prolog may also have a domains section. In this section the programmer can decline the type of each object.

Examples:

Clauses Section- likes(tom, anna)
Predicates Section-likes(boy, girl)
Domains Section-boy, girl = symbol

It is possible to omit the domains section by entering the data types of objects in the predicates section.
likes(symbol.symbol)
However, this might make the program harder to read especially if the predicate associates many objects.

Simple Program

domains

```
        disease,indication symbol
```

predicates

```
        symptom(disease,indication) clauses
        symptom(chicken_pox,high_fever).
        symptom(chicken_pox,chills).
        symptom(flu, chills).
        symptom(cold, mild body_ache)
        symptom(flu,severe body_ache).
        symptom(cold, runny nose)
        symptom(flu, runny_nose).
        symptom(flu, moderate cough)
```

Executing Simple Program

- Start Turbo Prolog
- Select the Edit mode
- Type in the program
- Exit the Editor using Esc.
- Save the program

- Select Run (which compiles the program for you in memory)

Once you have followed these steps you will see the following prompt in the Dialog Panel:

Goal:Using a Prolog program is essentially about asking questions. To ask the executing Prolog program a question you specify the Goal.

Ex-Goal: symptom(cold,runny_nose) True

Goal: Turbo Prolog will respond with True and prompt for another goal.

Possible outcomes of specifying a goal:

1. The goal will succeed; that is, it will be proven true
2. The goal will fail; Turbo Prolog will not be able to match the goal with any facts in the prograrm.
3. The execution will fail because of an error in the program.

Execution is a matching process. The program attempts to match the goal with one of the clauses in the clauses section beginning with the first clause. If it does find a complete match, the goal succeeds and True is displayed. In Prolog, False indicates a failure to find a match using the current database-not that the goal is untrue.

Variables Revisited

Variables are used in a clause or goal to specify an unknown quantity. Variables enable the user toask more informative questions. For example, if we wanted to know for which diseases, runny_nose was a symptom-type in

Goal: symptom(Disease, runny nose)
Turbo Prolog will respond
Disease = cold
Disease = f1u
2 Solutions

Goal:

To find the two solutions Turbo Prolog began at the start of the clauses section and tried to match the goal clause to one ot the clauses. When a match succeeded, the values of the variables for the, successful match was displayed. Turbo prolog continued this process until it had tested all predicates for a match with the specified goal.
If you wish Prolog to ignore the value of one or more arguments when determining a goal's failure or success then you can use the anonymous variable ”_” (underscore character).

Ex
Goal:Symptom chills).True Goal

Matching

Two facts match if their predicates are the same, and if their corresponding arguments are the same. When trying to match a goal that contains an uninstantiated variable as an argument, Prolog will allow that argument to match any other argument in the same position in the fact.
If a variable has a value associated with a value at a particular time it is instantiated otherwise it is uninstantiated.

Heuristics

- A method to help solve a problem, commonly informal.

- It is particularly used for a method that often rapidly leads to a solution that is usually reasonably' close to the best possible answer.

Heuristics are “rules of thumb", educated guesses, intuitive judgments or simply common sense.

PRACTICAL 2: WRITE A PROLOG PROGRAM OF PERSONS HOBBIES.

```
hobby(Jaydeep, dance).
hobby(Jiten, sports).
Hobby(Jivraj, sports)
hobby(Rishabh, game).

start:
Write(' Enter Person Name"),nl,
read().
hobby(X,Y),
write("Hobby is ),write(Y).
```

Output:

```
?-start.
Enter Person Name
|: Jivraj.
Hobby is sports
true.
```

PRACTICAL 3: WRITE PROLOG PROGRAM TO CHECK SYMPTOMS OF A DISEASE.

```
symptom(amit,fever)
symptom(amit.rash).
symptom(amit.headache).
symptom(amit.runny_nose).

symptom(kaushal,chills).
symplom(Kaushal,fever).
symptom(kaushal, headache).

symptom(dipen,runny_nose).
Symptom(dipen,rash).
Symptom(dipen,flu).

medicine(measels,paracetemol).
medicine(german_measels,vicks).
medicine(flu,crocin).
medicine(pain, 400).

hypothesis(Patient.meases):-
        symptom(Patient, fever),
        symptom(Patient, cough),
        symptom(Patient,.conjunctivitis),
        symptom(Patient.rash).

hypothesis(Patient,german_measels):-
        symptom(Patient,fever)
        symptom(Patient.headache),
        symptom(Patient,runny_nose),
        symptom(Patient,.rash).

hypothesis(Patient,pain):-
        symptom(Patient,fever),
        symptom(Patient,headache),

        symptom(Patient,chills).

hypothesis(Patient,flu):-
        symptom(Patient,fever)
        symptom(Patient.headache),
        symptom(Patient,body_pain),
        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).


Start:-

write(Enter Patient name="),nl,
read(Patient),
hypothesis(Patient,X),
write(You have'),write(X),nl,
medicine(X, Y),
write('Please Take this Medicine= ),write(Y),nl.
```

Output:

```
?-start.
Enter Patient name=
|:amit.
You have german_measels
Please Take this Medicine= vicks
true.
```

PRACTICAL 4: WRITE PROLOG PROGRAM TO FIND FACTORIAL OF NUMBER.

Description/ Program:

```
Predicates

factorial(integer,integer)
factorial(integer) Clauses

/*Base case, 0!=1*/
factorial(0,X):-
X=1


/*recursion for factorial*/

factorial(N,X):-
NN-=N-1,
factorial(NN,X!),
X=X1*N.

One argument function"/
factorial(N):-
factorial(N,X),
write(X).
```

Goal:

```
factorial(50).
120Yes
```

PRACTICAL 5: WRITE PROLOG PROGRAM TO FIND SUM OF ALL THE NUMBERS IN A GIVEN LIST.

Description/ Program:

```
domains

        list=integer*

predicates

        findsum(list)
        sum(list,integer)

clauses

findsum(L):-     sum(L,Sum
                 write("\nSum Or Given List :"Sum).

sum([],0).

  sum([XITail],Sum):-
sum(Tail,Temp),
Sum=TemptX.
```

Goal:

```
findsum([1,2,3,4,5]).
Sum Of Given List
15
Yes
```

Goal:

```
findsum([1,2,3,4,5,6,7,8,9,10]))
Sum Of Given List:55
Yes
```

PRACTICAL 6: WRITE PROLOG PROGRAM FOR APPENDING AND REVERSING A LIST.

Description/ Program:

Append list

```
Domains
        list-symbol*

predicates
        con(list,list,list)

clauses
        con([],LI,L1).
    con([XITail].L2,[X|Taill):
con(Tail,L2,Tail1).
```

Reverse List:

```
Domains
        list-integer*

predicates
        reverse_list(list,list)
        reverse(list,list,list)

clauses
        Revers_ list(lnputlist,outputlist):-
        reverse(Inputlist,[].Outputlist).
        reverse([],0utputlist,Outputlist).

        reverse([Head|Tail], List l,List2):-
        reverse(Tail,[Head|Listl],List2).
```

Goal:

```
con([a,b.eld,e].ConcatList)
ConcatList-[a","b","c","d","e"] |
Solution.
```

Goal:

```
reverse list([1,2,3
}.X) X-[3.2.1] 1
Solution
```

PRACTICAL 7: WRITE PROLOG PROGRAM TO SOLVE TOWER OF HANOI PROBLEM.

Description/ Program:

```
DOMAINS

        loc = right; middle; left

PREDICATES
hanoi(integer)
move(integer,loc.loc,loc)
intom(loc.loc)

CLAUSES
hanoi(N):-
        move(N,left,middle.right).

move(,A,,C):-
inform(A.C).!.
move(N,A,B,C):-   N1=N-1
move(NI,A,C.B),
inform(A,C),.move(NI,B,A,C).
inform(Locl, Loc2):-nl,
write("Move a disk from ", Locl,
" to ", Loc2).
```

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
```

PRACTICAL 8: WRITE A PROGRAM To IMPLEMENT DFS (FOR 8 PUZZLE PROBLEM OR WATER JUG PROBLEM OR ANY AI SEARCH PROBLEM)

Description/Algorithm:

If we want to go from Point A to Point B, you are employing some kind of search. For a direction finder, going from Point A to Point B literally means finding a path between where you are now and your intended destination. For a chess game, Point A to Point B might be two points between its current position and ts position S moves from now. For a,genome sequence, Points A and B could be a link between two DNA Sequences. As you can tell, going from Point A to Point B is different for every situation. If there is a vast amount of interconnected data, and you are trying to find a relation between few such pieces of data, you would use search. In this tutorial, you will learn about two forms of searching, depth first and breadth first. Searching falls under Artificial Intelligence (Al). A major goal of Al is to give computers the ability to think, or in other words, mimic human behaviour.Your goal, then, is to take a complicated task and convert into simpler steps that your computer can handle.

That conversion from something complex to something simple is what this tutorial is primarily about.

Learning how to use two search algorithms is just a welcome side-etfect.

Program (For water jug problem):

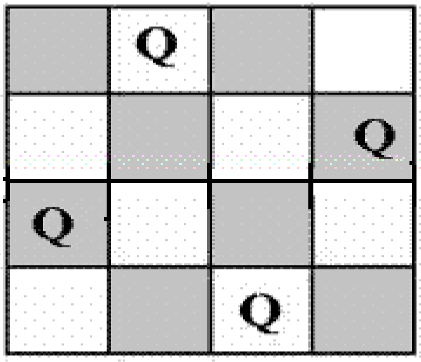
```
min(X,YX):-X<Y,!.
minY,Y

rev(L.R):-revacc(L].R).
revacc([].R,R):-!.
revacc([H|T|.A,R):-revacc(T[HA].R).

%Solve water jug problem using DFS
%X,Y are initial contents, Nx,Ny are final contents of jugl of capacity and jug2 of capacity My
respectively after pouring from jugl into jug2
chkX(_.My,.Y,Nx,Ny):           X>0,YMy.Ey is My-Y,min(X,Ey,P)
                        Nx is X-P,Ny is Y+P.

%Given 3 jugs of capacities Mx,My.Mz and filled with A, Y,L units ot a liquid
respectively.give steps so that finally they contain Fx.Fy.Fz units of the liquid respectively.
jug(Mx,My,Mz, X, Y,Z,Fx,Fy, Fz):-
jug(Mx,X,My, Y,Mz,Z.Fx.Fy.Fz.[1Initially'}). jug(Fx. Fy,Fz,Fx,Fy,Fz,T.R):-
,rev[Fx,Fy,Fz],[Fx,Fy,Fz||T].TR).rev(['Finally|R),.RR),display(TR,RR).
jug(Mx,X,My, Y, Mz,Z,Fx.Fy.Fz,1,R):-chk(Mx,X,My, Y,NX,Ny),not(member([Nx,Ny.ZJ.1))
jug(Mx,Nx,My,Ny, Mz,Z,Fx,Fy,.lFz,|[X, Y,Z|ITPour liquid from jugl into jug2|R]).
jug(Mx,X, My,Y,Mz,Z,Fx,Fy,Fz,1,R):-chk(Mx,.X,Mz,Z,Nx,N2), not(member([Nx, Y,Nz].T)
jug(Mx,Nx,My, Y,Mz,Nz,Fx,Fy,Fz,||X, Y,Z|TIPour liquid from jugl into jug3|R).
jug(Mx,X,My, Y,Mz,Z,Fx,Fy,Fz,1TR):-chk(My, Y,Mz,Z,Ny,N2),not(member((X,Ny.Nz].T))
jug(Mx,X,My,Ny, Mz,Nz,Fx,Fy,Fz,[|X, Y,ZJ|T],TPour liquid from jug2 into jug3|R).
jug(Mx.X.My, Y.MzZ,Fx.Fy.Fz.T.R):-chk(My.Y,Mx,X.Ny.Nx),not(member([Nx,Ny,Z],T)
jug(Mx.Nx.My.Ny.Mz.Z.Ex.Fy.Fz,[X.Y.Z|IT],['TPour liquid from jug2 into jugl"R).
jug(NIx.X.My. Y.Mz.Z.Fx.Fy.Fz.T.R):-chk(Mz.7,Mx.X.Nz,Nx),not(member([Nx, Y,Nz],1)
jug(Mx.Nx.My. Y.MMz.Nz.Fx.Fy.Fz.[[X. Y.Z||T|.Pour liquid from Jug5 into Jug)
Jug(Mx.X.My. Y.M7.7.Fx.Fy.Fz,T.R):-chk(Mz,Z.My.Y,Nz,Ny).not(member([X,Ny,NZ],1))
Jug(Mx.X.My.Ny.Mz.Nz,Fx.Fy,Fz.[X,Y,Z]|T],[Pour liquid from jug3 into jug2R).

display([]-0):-
display([T 1|T],[RIR):-write(RI),rite(: "),write(T),nl,display(T,R).
```

Output:

```
?- jug(8,5,3,8,0,0,4,4,0).
Initially: (8,0,0]
Pour liquid from jugl into jug2 : 13,5,0]
Pour liquid from jugl into jug3: [0,5,3]
Pour Liquid from jug2 into jugl [5,0,3]
Pour liquid from jug3 into jug2
Pour liquid from jugl into jug3
Pour liquid from jug3 nto jug2
Pour liquid from jug2 into jugl
Pour liquid from jug3 into jug2
Pour Liquid from Jugl 1nto Jug3: [4,1,3]
Pour liquid from jug3 lnto jug2 [4,4,0]
Finally: [4,4,0]
True
```

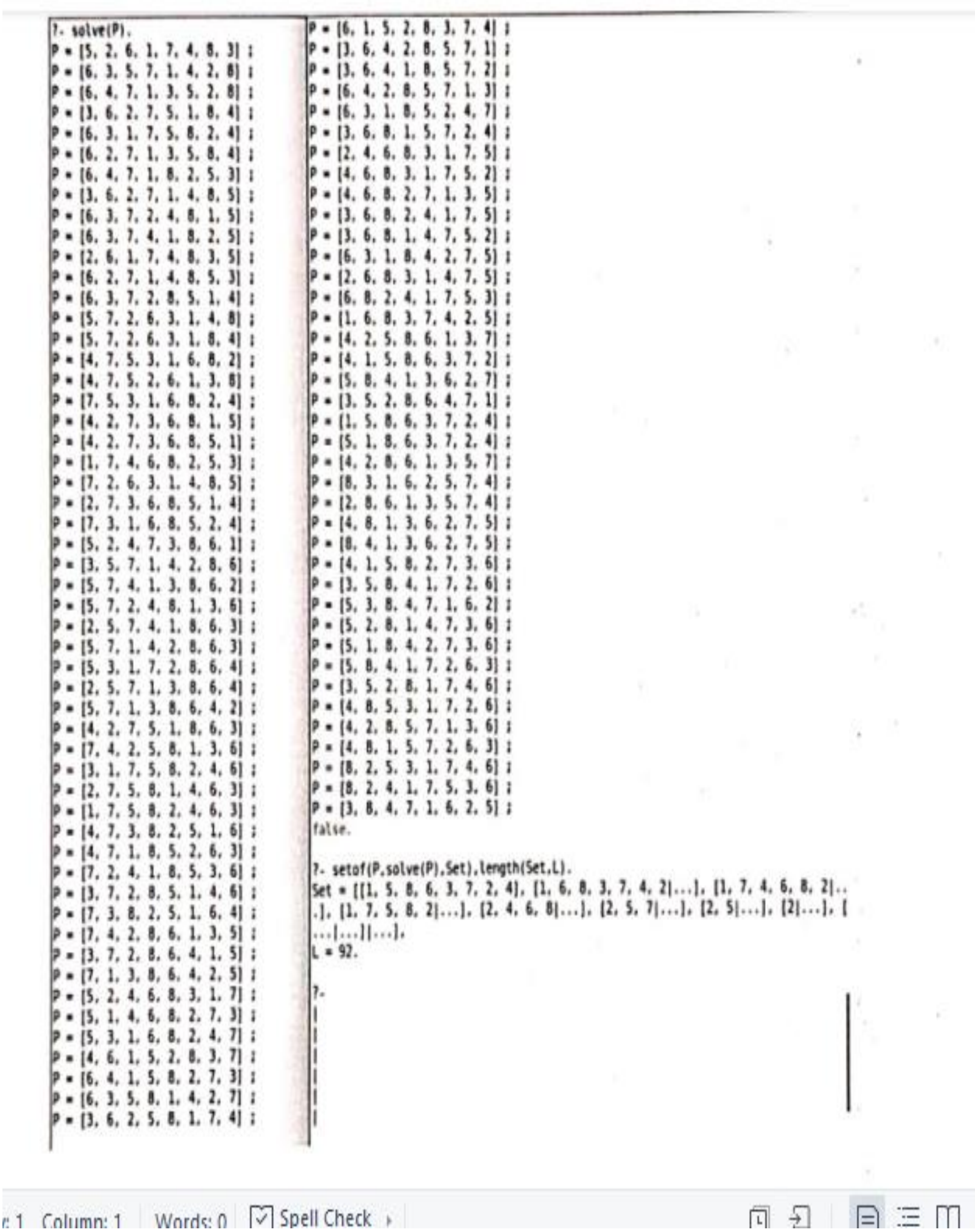
PRACTICAL 9: WRITE A PROGRAM TO SOLVE N-QUEENS PROBLEM USING PROLOG.

Description/Algorithm:



```
permXIY1Z)-perm(Y,W), takeout(X,Z, W)
perm([], []).

takeout(X,[XR],R).
takeout(X,[FIR],LFIS) :
takeout(X, R,S).

solve(P):
perm([1,2,3,4,5,6,7,8].P),
combine(1,2,3,4,5,6,7,8J.P,S,D)

all_dift (S),
all_dif(D).

combine([X1|X],[YI|Y].[S1|S],[D1|D
]):- Si is X1 +Y1, DI is X1
Y1,combine(X, Y,S,D).
combine([], [], [], []).

all _diff([X|Y]) :- \+member(X, Y), all_diff(Y).
all _diff(([X]).
```

Output:-



```
?- solve(P).
P = [5, 2, 6, 1, 7, 4, 8, 3] ;
P = [6, 3, 5, 7, 1, 4, 2, 8] ;
P = [6, 4, 7, 1, 3, 5, 2, 8] ;
P = [3, 6, 2, 7, 5, 1, 8, 4] ;
P = [6, 3, 1, 7, 5, 8, 2, 4] ;
P = [6, 2, 7, 1, 3, 5, 8, 4] ;
P = [6, 4, 7, 1, 8, 2, 5, 3] ;
P = [3, 6, 2, 7, 1, 4, 8, 5] ;
P = [6, 3, 7, 2, 4, 8, 1, 5] ;
P = [6, 3, 7, 4, 1, 8, 2, 5] ;
P = [2, 6, 1, 7, 4, 8, 3, 5] ;
P = [6, 2, 7, 1, 4, 8, 5, 3] ;
P = [6, 3, 7, 2, 8, 5, 1, 4] ;
P = [5, 7, 2, 6, 3, 1, 4, 8] ;
P = [5, 7, 2, 6, 3, 1, 8, 4] ;
P = [4, 7, 5, 3, 1, 6, 8, 2] ;
P = [4, 7, 5, 2, 6, 1, 3, 8] ;
P = [7, 5, 3, 1, 6, 8, 2, 4] ;
P = [4, 2, 7, 3, 6, 8, 1, 5] ;
P = [4, 2, 7, 3, 6, 8, 5, 1] ;
P = [1, 7, 4, 6, 8, 2, 5, 3] ;
P = [7, 2, 6, 3, 1, 4, 8, 5] ;
P = [2, 7, 3, 6, 8, 5, 1, 4] ;
P = [7, 3, 1, 6, 8, 5, 2, 4] ;
P = [5, 2, 4, 7, 3, 8, 6, 1] ;
P = [3, 5, 7, 1, 4, 2, 8, 6] ;
P = [5, 7, 4, 1, 3, 8, 6, 2] ;
P = [5, 7, 2, 4, 8, 1, 3, 6] ;
P = [2, 5, 7, 4, 1, 8, 6, 3] ;
P = [5, 7, 1, 4, 2, 8, 6, 3] ;
P = [5, 3, 1, 7, 2, 8, 6, 4] ;
P = [2, 5, 7, 1, 3, 8, 6, 4] ;
P = [5, 7, 1, 3, 8, 6, 4, 2] ;
P = [4, 2, 7, 5, 1, 8, 6, 3] ;
P = [7, 4, 2, 5, 8, 1, 3, 6] ;
P = [3, 1, 7, 5, 8, 2, 4, 6] ;
P = [2, 7, 5, 8, 1, 4, 6, 3] ;
P = [1, 7, 5, 8, 2, 4, 6, 3] ;
P = [4, 7, 3, 8, 2, 5, 1, 6] ;
P = [4, 7, 1, 8, 5, 2, 6, 3] ;
P = [7, 2, 4, 1, 8, 5, 3, 6] ;
P = [3, 7, 2, 8, 5, 1, 4, 6] ;
P = [7, 3, 8, 2, 5, 1, 6, 4] ;
P = [7, 4, 2, 8, 6, 1, 3, 5] ;
P = [3, 7, 2, 8, 6, 4, 1, 5] ;
P = [7, 1, 3, 8, 6, 4, 2, 5] ;
P = [5, 2, 4, 6, 8, 3, 1, 7] ;
P = [5, 1, 4, 6, 8, 2, 7, 3] ;
P = [5, 3, 1, 6, 8, 2, 4, 7] ;
P = [4, 6, 1, 5, 2, 8, 3, 7] ;
P = [6, 4, 1, 5, 8, 2, 7, 3] ;
P = [6, 3, 5, 8, 1, 4, 2, 7] ;
P = [3, 6, 2, 5, 8, 1, 7, 4] ;
```

```
P = [6, 1, 5, 2, 8, 3, 7, 4] ;
P = [3, 6, 4, 2, 8, 5, 7, 1] ;
P = [3, 6, 4, 1, 8, 5, 7, 2] ;
P = [6, 4, 2, 8, 5, 7, 1, 3] ;
P = [6, 3, 1, 8, 5, 2, 4, 7] ;
P = [3, 6, 8, 1, 5, 7, 2, 4] ;
P = [2, 4, 6, 8, 3, 1, 7, 5] ;
P = [4, 6, 8, 3, 1, 7, 5, 2] ;
P = [4, 6, 8, 2, 7, 1, 3, 5] ;
P = [3, 6, 8, 2, 4, 1, 7, 5] ;
P = [3, 6, 8, 1, 4, 7, 5, 2] ;
P = [6, 3, 1, 8, 4, 2, 7, 5] ;
P = [2, 6, 8, 3, 1, 4, 7, 5] ;
P = [6, 8, 2, 4, 1, 7, 5, 3] ;
P = [1, 6, 8, 3, 7, 4, 2, 5] ;
P = [4, 2, 5, 8, 6, 1, 3, 7] ;
P = [4, 1, 5, 8, 6, 3, 7, 2] ;
P = [5, 8, 4, 1, 3, 6, 2, 7] ;
P = [3, 5, 2, 8, 6, 4, 7, 1] ;
P = [1, 5, 8, 6, 3, 7, 2, 4] ;
P = [5, 1, 8, 6, 3, 7, 2, 4] ;
P = [4, 2, 8, 6, 1, 3, 5, 7] ;
P = [8, 3, 1, 6, 2, 5, 7, 4] ;
P = [2, 8, 6, 1, 3, 5, 7, 4] ;
P = [4, 8, 1, 3, 6, 2, 7, 5] ;
P = [8, 4, 1, 3, 6, 2, 7, 5] ;
P = [4, 1, 5, 8, 2, 7, 3, 6] ;
P = [3, 5, 8, 4, 1, 7, 2, 6] ;
P = [5, 3, 8, 4, 7, 1, 6, 2] ;
P = [5, 2, 8, 1, 4, 7, 3, 6] ;
P = [5, 1, 8, 4, 2, 7, 3, 6] ;
P = [5, 8, 4, 1, 7, 2, 6, 3] ;
P = [3, 5, 2, 8, 1, 7, 4, 6] ;
P = [4, 8, 5, 3, 1, 7, 2, 6] ;
P = [4, 2, 8, 5, 7, 1, 3, 6] ;
P = [4, 8, 1, 5, 7, 2, 6, 3] ;
P = [8, 2, 5, 3, 1, 7, 4, 6] ;
P = [8, 2, 4, 1, 7, 5, 3, 6] ;
P = [3, 8, 4, 7, 1, 6, 2, 5] ;
false.

?- setof(P,solve(P),Set),length(Set,L).
Set = [[1, 5, 8, 6, 3, 7, 2, 4], [1, 6, 8, 3, 7, 4, 2|...], [1, 7, 4, 6, 8, 2|..
.], [1, 7, 5, 8, 2|...], [2, 4, 6, 8|...], [2, 5, 7|...], [2, 5|...], [2|...], [
...|...]|...],
L = 92.

?-
|
|
|
|
|
|
```

PRACTICAL 10: WRITE A PROGRAM TO SOLVE TRAVELLING SALESMAN PROBLEM USING PROLOG.

Description/Algorithm:

TSP Problem
A traveling salesman has to travel through a bunch of cities, in sucha way that the expenses on traveling are minimized. This is the infamous Traveling Salesman Problem (aka TSP) problem it belongs to a family of problems, called NP-complete problem. It is conjectured that all those problems requires exponential time to solve them. In our case, this means that to find the optimal solution you have to go through all possible routes, and the numbers of routes increases exponential with the numbers of cities.

```
road(hjd,bhuj.27).
road(bhuj.anjar,40).
road(anjar,adipur, 15).
road(adipur gandhidham,8).

start:-
        write( Enter source city" ),nl,
        read(Source),nl,
        Write( Enter destination city"),nl,
        read(Dest),nl,

        route(Source,Dest).

route(X, Y):-
        road(X,Y,D3),
        write (total distance+"),write(D3).

route(X, Y):-
        write(X),nl,
        road X.R,DI),
        write(R).nl,
        road(R,Y,D2),
        write(Y),nl,

        Total-D1+D2,
        write("total distance"), write(Total).
```

Output:

```
?-start                                   hjd
Enter source city                         bhuj
: hjd.                                    anjar
                                          total distance27+40
Enter destination city
:anjar.
```