PROLOG Lab for AI

Nagashree N
Asst Prof
Dept of CSE
<table>
<thead>
<tr>
<th>S.NO</th>
<th>PROGRAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Study of PROLOG. Write the following programs using PROLOG</td>
</tr>
<tr>
<td>2</td>
<td>Write a program to solve 8 queens problem</td>
</tr>
<tr>
<td>3</td>
<td>Solve any problem using depth first search.</td>
</tr>
<tr>
<td>4</td>
<td>Solve any problem using best first search.</td>
</tr>
<tr>
<td>5</td>
<td>Solve 8-puzzle problem using best first search</td>
</tr>
<tr>
<td>6</td>
<td>Solve Robot (traversal) problem using means End Analysis</td>
</tr>
<tr>
<td>7</td>
<td>Solve traveling salesman problem.</td>
</tr>
</tbody>
</table>
Introduction to Prolog

• **PROLOG** is a simple, yet powerful programming language, based on the principles of first-order predicate logic. The name of the language is an acronym for the French 'PROgramma-tion en LOGique'. About 1970, **PROLOG** was designed by A. Colmerauer and P.
• Prolog is the most commonly used logic
  - Simple syntax
• PROgramming in LOGic
• High-level interactive language.
• Logic programming language.
The use of Prolog

- Prolog is used in artificial intelligence applications such as:

  - Natural language interfaces,
  - automated reasoning systems and
  - Expert systems: usually consist of a data base of facts and rules and an inference engine, the run time system of Prolog provides much of the services of an inference engine.
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 for 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.
- A program consists 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.
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 of 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

- **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.

- **Domain Section**
  Programmer can define the type of each object
Examples

Clauses Section – \textit{likes}(tom, anna).

Predicates Section – \textit{likes}(boy, girl)
Domains Section – boy, girl \textit{=} symbol
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:
  • 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 program.
3. The execution will fail because of an error in the program.

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 program.
3. The execution will fail because of an error in the program.
Program to demonstrate a simple prolog program.

predicates
    like(symbol,symbol)
    hate(symbol,symbol)

clauses
    like(sita,ram).
    like(x,y).
    like(a,b).

    hate(c,d).
    hate(m,n).
    hate(f,g).
Goal: like(sita, ram)
Yes
Goal: like(a, X)
X=b
1 Solution
Goal: hate(c, X)
X=d
1 Solution
Goal: like(x, _) 
Yes
Goal: _