##### Basics related to Prolog

**Symbols**

Prolog expressions are comprised of the following truth-functional symbols, which have the same interpretation as in the predicate calculus.

|  |  |  |
| --- | --- | --- |
| **English** | **Predicate Calculus** | **PROLOG** |
| **And** | ^ | , |
| **Or** | **V** | ; |
| **If** | → | :- |
| **Not** | ~ | **Not** |

Variables begin with an uppercase letter. Predicate names, function names, and the names for objects must begin with a lowercase letter. Rules for forming names are the same as for the predicate calculus. mother\_of

male female

greater\_than socrates

##### Facts

A **fact** is a predicate expression that makes a declarative statement about the problem domain. Whenever a variable occurs in a Prolog expression, it is assumed to be **universally quantified**. Note that all Prolog sentences must end with a period.

likes(john, susie). /\* John likes Susie \*/ likes(X, susie). /\* Everyone likes Susie \*/ likes(john, Y). /\* John likes everybody \*/

likes(john, Y), likes(Y, john). /\* John likes everybody and everybody likes John \*/ likes(john, susie); likes(john,mary). /\* John likes Susie or John likes Mary \*/ not(likes(john,pizza)). /\* John does not like pizza \*/

likes(john,susie) :- likes(john,mary)./\* John likes Susie if John likes Mary.

##### Rules

A **rule** is a predicate expression that uses logical implication (:-) to describe a relationship among facts. Thus, a Prolog rule takes the form

left\_hand\_side :- right\_hand\_side .

* + This sentence is interpreted as**: *left\_hand\_side if right\_hand\_side*.**
  + The **left\_hand\_side** is restricted to a **single, positive, literal**, which means it must consist of a positive atomic expression.
  + It cannot be negated and it cannot contain logical connectives.

This notation is known as a **Horn clause**. In Horn clause logic, the left-hand side of the clause is the conclusion, and must be a single positive literal. The right-hand side contains the premises. The Horn clause calculus is equivalent to the first-order predicate calculus.

##### Examples of valid rules:

friends(X,Y) :- likes(X,Y),likes(Y,X). /\* X and Y are friends if they like each other \*/ hates(X,Y) :- not(likes(X,Y)). /\* X hates Y if X does not like Y. \*/

enemies(X,Y) :- not(likes(X,Y)),not(likes(Y,X)). /\* X and Y are enemies if they don't like each other \*/

##### Examples of invalid rules:

left\_of(X,Y) :- right\_of(Y,X) /\* Missing a period \*/ likes(X,Y),likes(Y,X) :- friends(X,Y). /\* LHS is not a single literal \*/ not(likes(X,Y)) :- hates(X,Y). /\* LHS cannot be negated \*/

1. **Study of Prolog : Write simple facts for the statements and querying it. Sentences:**

Human is mortal India capital is Delhi

Telangana capital is Hyderabad Maharashtra capital is Mumbai Tamilnadu capital is Chennai Karnataka capital is bangalore

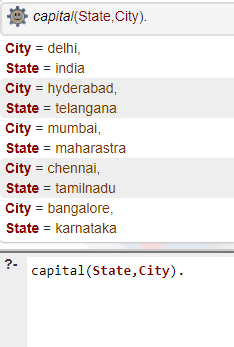
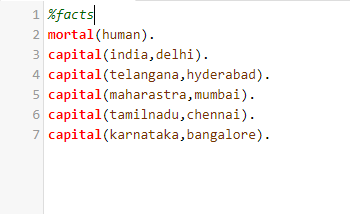
**Facts:**

mortal(human). capital(india,delhi). capital(telangana,hyderabad). capital(maharashtra, mumbai). capital(tamilnadu,chennai). capital(karnataka, bangalore).

**Queries:**

capital(State,City)

**Output:**



##### Simple prolog program with facts, rules and queries. Program:

% Facts

* state(telangana).
* state(maharashtra).
* state(tamilnadu).
* state(karnataka).

% Facts

* capital(telangana,hyderabad).
* capital(maharashtra, mumbai).
* capital(tamilnadu,chennai).
* capital(karnataka, bangalore).

% Facts

* language(telangana,telugu).
* language(maharashtra,marati).
* language(tamilnadu,tamil).
* language(maharashtra,marati).
* language(karnataka,kannada).

% Rules

* language(X,english):-state(X).
* language(X,hindi):-state(X).

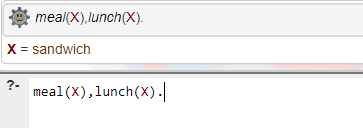
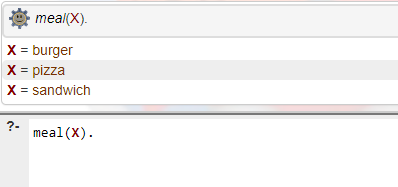
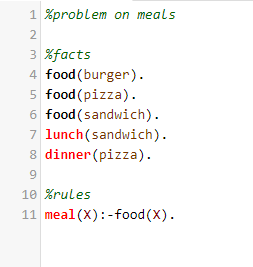
%Queries

* about\_city(Z):-state(X),capital(X,Z),language(X,Y),write(Z),write(" is the capital of "),write(X), write(" and the language is "),write(Y),nl,fail.

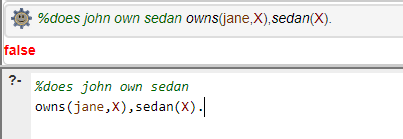
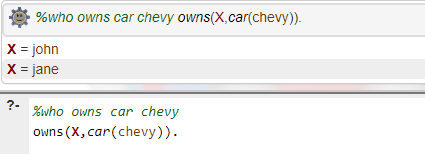
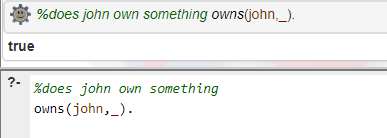
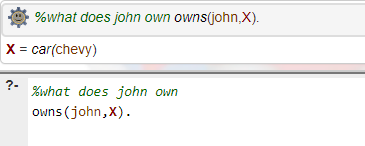
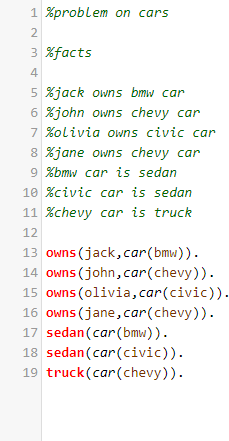
##### Output:



**Simple prolog program with facts, rules and queries.**



##### Simple prolog program with facts, rules and queries.



1. **Study of Prolog: Write a program for Family-tree.**

##### Facts

* + male(jack).
  + male(oliver).
  + male(ali).
  + male(james).
  + male(simon).
  + male(harry).
  + female(helen).
  + female(sophie).
  + female(jess).
  + female(lily).
  + parent\_of(jack,jess).
  + parent\_of(jack,lily).
  + parent\_of(helen, jess).
  + parent\_of(helen, lily).
  + parent\_of(oliver,james).
  + parent\_of(sophie, james).
  + parent\_of(jess, simon).
  + parent\_of(ali, simon).
  + parent\_of(lily, harry).
  + parent\_of(james, harry).

##### Rules

* + father\_of(X,Y):- male(X), parent\_of(X,Y).
  + mother\_of(X,Y):- female(X), parent\_of(X,Y).
  + grandfather\_of(X,Y):- male(X), parent\_of(X,Z), parent\_of(Z,Y)
  + grandmother\_of(X,Y):- female(X), parent\_of(X,Z), parent\_of(Z,Y).
  + sister\_of(X,Y):- female(X), mother\_of(M, Y), mother\_of(M,X), X \= Y.
  + brother\_of(X,Y):- male(X), mother\_of(M, Y), mother\_of(M,X),X \= Y.
  + ancestor\_of(X,Y):- parent\_of(X,Y).
  + ancestor\_of(X,Y):- parent\_of(X,Z), ancestor\_of(Z,Y).

##### Queries

* + Who is father of jess
  + Who is grand father of simon
  + Who is sister of jess

##### Output:

