

Practical No.1

Title: Implementation of Logic programming

Aim: Understanding basics of prolog and implementation of prolog to solve tower of hanoi problem, water jug problem, tic-tac-toe problem and 8- Puzzle Problem.

Introduction:

What is Prolog :-

Prolog (programming in logic) is one of the most widely used programming languages in artificial intelligence research. As opposed to imperative languages such as C or Java (the latter of which also happens to be object-oriented) it is a declarative programming language. That means, when implementing the solution to a problem, instead of specifying how to achieve a certain goal in a certain situation, we specify what the situation (rules and facts) and the goal (query) are and let the Prolog interpreter derive the solution for us. Prolog is very useful in some problem areas, such as artificial intelligence, natural language processing, databases, . . . , but pretty useless in others, such as graphics or numerical algorithms.

Example : -

male(james1).

male(charles1).

male(charles2).

male(james2).

male(george1).

female(catherine).

female(elizabeth).

female(sophia).

parent(charles1, james1).

parent(elizabeth, james1).

parent(charles2, charles1).

parent(catherine, charles1).

parent(james2, charles1).

parent(sophia, elizabeth).

parent(george1, sophia).

mother(M,X):- parent(X,M),female(M),write(M),write(' is Mother of '),write(X),nl.

father(F,X):- parent(X,F),male(F).

sibling(S1,S2):- parent(S1,X), parent(S2,X).

grandfather(G,X) :- parent(Y,G),parent(X,Y).

Exercise -**Implementation:**

1. Write a Prolog predicate `reverse_list/2` that works like the built-in predicate `reverse/2` (without using `reverse/2`).

Program:

% Base case: Reversing an empty list results in an empty list.

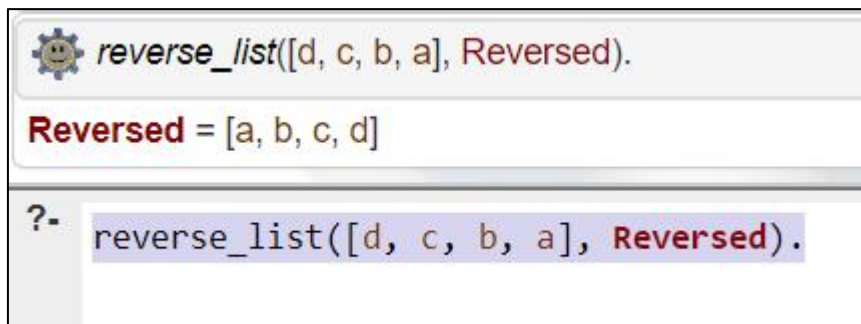
```
reverse_list([], []).
```

% Recursive case: To reverse a non-empty list, append the reversed tail to the head.

```
reverse_list([Head|Tail], Reversed) :-
```

```
    reverse_list(Tail, ReversedTail),
```

```
    append(ReversedTail, [Head], Reversed).
```

Output:

The screenshot shows a Prolog interpreter window with a gear icon. The first line is the query `reverse_list([d, c, b, a], Reversed).`. The second line shows the result `Reversed = [a, b, c, d]`. The third line shows the prompt `?- reverse_list([d, c, b, a], Reversed).` with the text highlighted in blue.


2. Write a Prolog predicate distance/3 to calculate the distance between two points in the 2-dimensional plane. Points are given as pairs of coordinates.

Program:

distance((X1, Y1), (X2, Y2), Distance) :-

Distance is sqrt((X2 - X1) ** 2 + (Y2 - Y1) ** 2).

Output:



The screenshot shows a Prolog interpreter window with a title bar containing a gear icon and standard window controls. The main text area displays the query `distance((-2, 1), (5, -3), Distance).` in red. Below the query, the result `Distance = 8.06225774829855` is shown in red. At the bottom, a prompt `?-` is followed by the same query `distance((-2, 1), (5, -3), Distance).`

3. Write a prolog program to solve Tower of hanoi Problem.**Program:**

% Predicate to solve Tower of Hanoi problem

hanoi(N) :-

 move(N, left, middle, right).

% Predicate to move disks

move(1, Source, _, Destination) :-

 write('Move top disk from '),

 write(Source),

 write(' to '),

 write(Destination),

 nl.

move(N, Source, Auxiliary, Destination) :-

 N > 1,

 M is N - 1,

 move(M, Source, Destination, Auxiliary),

 move(1, Source, _, Destination),

 move(M, Auxiliary, Source, Destination).

Output:

```
hanoi(3).  
Move top disk from left to right  
Move top disk from left to middle  
Move top disk from right to middle  
Move top disk from left to right  
Move top disk from middle to left  
Move top disk from middle to right  
Move top disk from left to right  
true  
Next 10 100 1,000 Stop  
?- hanoi(3).
```

4. Write a Prolog predicate fibonacci/2 to compute the nth Fibonacci number.**Program:**

```
fibonacci(0, 0).
```

```
fibonacci(1, 1).
```

```
fibonacci(N, Fib) :-
```

```
    N > 1,
```

```
    N1 is N - 1,
```

```
    N2 is N - 2,
```

```
    fibonacci(N1, Fib1),
```

```
    fibonacci(N2, Fib2),
```

```
    Fib is Fib1 + Fib2.
```

Output: