**Name:** Kartik Jolapara **SAPID:** 60004200107

**DIV:** B/B1

# AI

## Exp6

**Aim:** Implementation of Wumpus World Program in Prolog.

## Theory:

Unlike many other programming languages, Prolog is intended primarily as a declarative programming language. In prolog, logic is expressed as relations (called Facts and Rules). The core heart of the prolog lies in the logic being applied.

Formulation or Computation is carried out by running a query over these relations. Prolog features are 'Logical variable', which means that they behave like uniform data structures, a backtracking strategy to search for proofs, a pattern-matching facility, mathematical variables, and input and out are interchangeable. Prolog is a declarative language that means we can specify what problem we want to solve rather than how to solve it. Prolog is used in some areas like databases, natural language processing, and artificial intelligence, but it is pretty useless in some areas like numerical algorithms or instance graphics.

Key Features of Prolog:

1. Unification: The basic idea is, can the given terms be made to represent the same structure.
2. Backtracking: When a task fails, prolog traces backward and tries to satisfy the previous task.
3. Recursion: Recursion is the basis for any search in the program.

Prolog provides an easy to build a database. Doesn't need a lot of programming effort. Pattern matching is easy. Search is recursion based. It has built-in list handling. Makes it easier to play with any algorithm involving lists. LISP (another logic programming language) dominates over prolog with respect to I/O features. Sometimes input and output is not easy.

Applications:

- Specification Language
- Robot Planning
- Natural language understanding
- Machine Learning
- Problem Solving
- Intelligent Database retrieval
- Expert System
- Automated Reasoning

**Code:**

```
/* Facts */ male(jack).

male(oliver).

male(ali).

male(james).

male(simon).

male(harry).

male(bhupesh).

male(ram).

female(helen).

female(sophie).

female(jess).
```

female(lily).

female(sita).

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).

parent_of(jack,bhupesh).

parent_of(helen,bhupesh).

parent_of(ram,helen).

parent_of(sita,helen).

parent_of(ram,sophie).

parent_of(sita,sophie).

/* Rules */ father_of(X,Y):-

male(X), parent_of(X,Y).

mother_of(X,Y):- female(X),

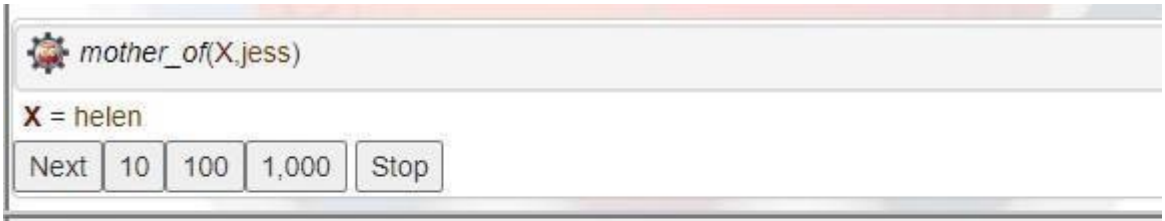
parent_of(X,Y).

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):- %(X,Y or Y,X)% female(X),

```
father_of(F,     Y), father_of(F,X),X
\=       Y.
sister_of(X,Y):-     female(X),
mother_of(M,                 Y),
mother_of(M,X),X    \=    Y.
aunt_of(X,Y):-       female(X),
parent_of(Z,Y),
sister_of(Z,X),!.
brother_of(X,Y):-  %(X,Y   or
Y,X)%                  male(X),
father_of(F,                 Y),
father_of(F,X),X     \=      Y.
brother_of(X,Y):-    male(X),
mother_of(M,                 Y),
mother_of(M,X),X    \=    Y.
uncle_of(X,Y):-
parent_of(Z,Y),
brother_of(Z,X).
ancestor_of(X,Y):-
parent_of(X,Y).
ancestor_of(X,Y):-
parent_of(X,Z),
ancestor_of(Z,Y).
```

**Output**:


father_of(X,jess)
X = jack
Next
10
100
1,000
Stop
?-
father_of(X,jess)




```
?- mother_of(X,jess)
```


```
?- grandfather_of(X,simon)
```


```
?- sister_of(X,jess)
```




**Conclusion**: Thus, we successfully implemented family tree in prolog.