EXPERIMENT-1

Aim- Introduction to artificial intelligence.

Theory-

Artificial intelligence (AI) is the simulation of human intelligence processes by machines, especially computer systems. These processes include learning (the acquisition of information and rules for using the information), reasoning (using rules to reach approximate or definite conclusions) and self-correction. Particular applications of AI include expert systems, speech recognition and machine vision.

AI can be categorized as either weak or strong. Weak AI, also known as narrow AI, is an AI system that is designed and trained for a particular task. Virtual personal assistants, such as Apple's Siri, are a form of weak AI. Strong AI, also known as artificial general intelligence, is an AI system with generalized human cognitive abilities. When presented with an unfamiliar task, a strong AI system is able to find a solution without human intervention.

AI programming using prolog:

Prolog is a logic programming language. It has important role in artificial intelligence. Unlike many other programming languages, Prolog is intended primarily as a declarative programming language. In prolog, logic is expressed as relations (called as Facts and Rules). Core heart of prolog lies at the logic being applied. Formulation or Computation is carried out by running a query over these relations.

Some definitions related to prolog:

1. **Symbols**

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

2. **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.

3. 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 .

4. **Queries**

The Prolog interpreter responds to queries about the facts and rules represented in its database. The database is assumed to represent what is true about a particular problem domain.

Advantages :

1. Easy to build database. Doesn't need a lot of programming effort.

2. Pattern matching is easy. Search is recursion based.

3. It has built in list handling. Makes it easier to play with any algorithm involving lists.

Disadvantages :

1. LISP (another logic programming language) dominates over prolog with respect to I/O features.

2. Sometimes input and output is not easy.

Applications :

Prolog is highly used in artificial intelligence(AI). Prolog is also used for pattern matching over natural language parse trees.

EXPERIMENT-2

Aim- Write a prolog program to find the rules for parent, child, male, female, son, daughter, brother, sister, uncle, aunt, ancestor given the facts about father and wife only.

Facts-

```
male(hari).

female(lina).

male(ram).

male(aman).

male(lishant).

female(sita).

female(geeta).

wife(geeta,aman).

wife(sita,lishant).

male(pradeep).

male(charles).

male(wasim).

parent(pradeep,charles).

parent(lina,charles).

parent(charles,hari).

parent(charles,wasim).

parent(aman,lishant).
```

Rules-

```
grandmother(GM,C) :-
        female(GM),
        mother(GM,P),
        parent(P,C).
```

```
mother(M,C) :-
        female(M),
        parent(M,C).
father(F,C) :-
        male(F),
        parent(F,C).
grandfather(GF,C) :-
        male(GF),
        parent(GF,X),
        parent(X,C).
brother(X,Y) :-
        male(X),
        parent(Z,X),
        parent(Z,Y).
sister(X,Y) :-
        female(X),
        parent(Z,X),
        parent(Z,Y).
sibling(X,Y) :-
        brother(X,Y).
sibling(X,Y) :-
        sister(X,Y).
uncle(X,Y) :-
        male(X),
        sibling(X,Z),
        parent(Z,Y).
aunt(X,Y) :-
        female(X),
        sibling(X,Z),
        parent(Z,Y).
```

Queries-

?- sibling(hari,X).
X = hari ;
X = wasim ;
false.

?- sibling(hari,X).
X = hari ;
X = wasim .

?- male(X).
X = hari ;
X = ram ;
X = aman ;
X = lishant ;
X = pradeep ;
X = charles ;
X = wasim.

?- uncle(ram,X).
false.

?- mother(lina,X).
X = charles.

?- grandfather(X,Y).
X = pradeep,
Y = hari ;
X = pradeep,
Y = wasim ;
false.

EXPERIMENT-3

Aim- To write a prolog program to define a predicate that checks if a number is positive.

Facts-

N>0.

Rules-

check_positive(N):-

N>0.

Queries-

?- check_positive (-9).
false.

?- check_positive (-3).
false.

?- check_positive (75).
true.

?- check_positive (100).
true.

?- check_positive (1).
true.

?- check_positive (-1).
False .

EXPERIMENT-4

Aim- To write a prolog program to define a predicate that checks if a number is odd or even.

Facts-

For odd numbers:-1 is mod(N,2).

For even numbers: - 0 is mod(N,2).

Rules-

even(N):-

    0 is mod(N,2).

odd(N):-

    1 is mod(N,2).

Queries-

?- even(42).
true.

?- even(436).
true.

?- even(59).
false.

?- odd(7).
true.

?- odd(3211).
true.

?- odd(20).
false.

EXPERIMENT-5

Aim- To write a prolog program to find factorial of a number.

Facts-

```
factorial(0,1).
```

Rules-

```
factorial(N,Z):-
    N>0,
    N1 is N-1,
    factorial(N1,Z1),
    Z is N*Z1.
```

Queries-

```
?- factorial(5,F).
F = 120.

?- factorial(4,F).
F = 24.

?- factorial(7,F).
F = 5040.

?- factorial(8,F).
F = 40320.

?- factorial(0,F).
F = 1.

?- factorial(2,F).
F = 2.
```

EXPERIMENT-6

Aim- To write a prolog program to define a predicate that takes a number as its first argument and return square of that number as its second argument.

Facts-

X is N*N.

Rules-

square(N,X):-

X is N*N.

Queries-

?- square(10,N).
N = 100.

?- square(6,N).
N = 36.

?- square(8,N).
N = 64.

?- square(50,N).
N = 2500.

?- square(67,N).
N = 4489.

?- square(88,N).
N = 7744.

EXPERIMENT-7

Aim- To write a prolog program to define a predicate that takes a number as its first argument and second argument as p and n and return as the value of third argument a number n^p.

Facts-

```
exponent(N,1,Y):-
  Y is N.
exponent(1,P,Y):-
  Y is 1.
```

Rules-

```
exponent(N,P,Y):-
  N>0,
  P1 is P-1,
  exponent(N,P1,Y1),
  Y is N*Y1.
```

Queries-

```
?- exponent(2,5,X).
X = 32 .

?- exponent(4,3,X).
X = 64 .

?- exponent(5,2,X).
X = 25 .

?- exponent(11,2,X).
X = 121 .

?- exponent(12,3,X).
X = 1728  .
```