## **Aim:**

# **Write a program to solve water jug problems using Prolog**

## **Solution :**

/\* Description:

"You are given two jugs, a 4-gallon one and a 3-gallon one. Neither have any measuring markers on it. There is a tap that can be used to fill the jugs with water. How can you get exactly 2 gallons of water into the 4-gallon jug?".

\*/

/\* Production Rules:-

R1: (x,y) --> (4,y) if x < 4

R2: (x,y) --> (x,3) if y < 3

R3: (x,y) --> (x-d,y) if x > 0

R4: (x,y) --> (x,y-d) if y > 0

R5: (x,y) --> (0,y) if x > 0

R6: (x,y) --> (x,0) if y > 0

R7: (x,y) --> (4,y-(4-x)) if x+y >= 4 and y > 0

R8: (x,y) --> (x-(3-y),y) if x+y >= 3 and x > 0

R9: (x,y) --> (x+y,0) if x+y =< 4 and y > 0

R10: (x,y) --> (0,x+y) if x+y =< 3 and x > 0

\*/

%database

visited\_state(integer,integer).

%predicates

state(integer,integer).

%clauses

state(2,0).

state(X,Y):- X < 4,

not(visited\_state(4,Y)),

assert(visited\_state(X,Y)),

write("Fill the 4-Gallon Jug: (",X,",",Y,") --> (", 4,",",Y,")\n"),

state(4,Y).

state(X,Y):- Y < 3,

not(visited\_state(X,3)),

assert(visited\_state(X,Y)),

write("Fill the 3-Gallon Jug: (", X,",",Y,") --> (", X,",",3,")\n"),

state(X,3).

state(X,Y):- X > 0,

not(visited\_state(0,Y)),

assert(visited\_state(X,Y)),

write("Empty the 4-Gallon jug on ground: (", X,",",Y,") --> (", 0,",",Y,")\n"),

state(0,Y).

state(X,Y):- Y > 0,

not(visited\_state(X,0)),

assert(visited\_state(X,0)),

write("Empty the 3-Gallon jug on ground: (", X,",",Y,") --> (", X,",",0,")\n"),

state(X,0).

state(X,Y):- X + Y >= 4,

Y > 0,

NEW\_Y = Y - (4 - X),

not(visited\_state(4,NEW\_Y)),

assert(visited\_state(X,Y)),

write("Pour water from 3-Gallon jug to 4-gallon until it is full: (", X,",",Y,") --> (", 4,",",NEW\_Y,")\n"),

state(4,NEW\_Y).

state(X,Y):- X + Y >=3,

X > 0,

NEW\_X = X - (3 - Y),

not(visited\_state(X,3)),

assert(visited\_state(X,Y)),

write("Pour water from 4-Gallon jug to 3-gallon until it is full: (", X,",",Y,") --> (", NEW\_X,",",3,")\n"),

state(NEW\_X,3).

state(X,Y):- X + Y>=4,

Y > 0,

NEW\_X = X + Y,

not(visited\_state(NEW\_X,0)),

assert(visited\_state(X,Y)),

write("Pour all the water from 3-Gallon jug to 4-gallon: (", X,",",Y,") --> (", NEW\_X,",",0,")\n"),

state(NEW\_X,0).

state(X,Y):- X+Y >=3,

X > 0,

NEW\_Y = X + Y,

not(visited\_state(0,NEW\_Y)),

assert(visited\_state(X,Y)),

write("Pour all the water from 4-Gallon jug to 3-gallon: (", X,",",Y,") --> (", 0,",",NEW\_Y,")\n"),

state(0,NEW\_Y).

state(0,2):- not(visited\_state(2,0)),

assert(visited\_state(0,2)),

write("Pour 2 gallons from 3-Gallon jug to 4-gallon: (", 0,",",2,") --> (", 2,",",0,")\n"),

state(2,0).

state(2,Y):- not(visited\_state(0,Y)),

assert(visited\_state(2,Y)),

write("Empty 2 gallons from 4-Gallon jug on the ground: (", 2,",",Y,") --> (", 0,",",Y,")\n"),

state(0,Y).

goal:-

makewindow(1,2,3,"4-3 Water Jug Problem",0,0,25,80),

state(0,0).

## **Output:**

% Goal:-

makewindow(1,2,3,"4-3 Water Jug Problem",0,0,25,80),

state(0,0).

+-----------------------------4-3 Water Jug Problem--------------------------+|

Fill the 4-Gallon Jug: (0,0) --> (4,0) Fill the 3-Gallon Jug: (4,0) --> (4,3)

Empty the 4-Gallon jug on ground: (4,3) --> (0,3) Pour all the water from 3-Gallon jug to 4-gallon: (0,3) --> (3,0)

Fill the 3-Gallon Jug: (3,0) --> (3,3) Pour water from 3-Gallon jug to 4-gallon until it is full: (3,3) --> (4,2)

Empty the 4-Gallon jug on ground: (4,2) --> (0,2) Pour all the water from 3-Gallon jug to 4-gallon: (0,2) --> (2,0)

### Write a Prolog program to remove the Nth item from a list.

**% Write a Prolog program to remove the Nth item from a list.**

**/\*delete a number in the list. \*/delte(1,[\_|T],T).delte(P,[X|Y],[X|R]):-**

**P1 is P-1,**

**delte(P1,Y,R).**

**% Output**

