Name of Student: Sunny Satish Halkatti		
Roll Number: 17 (B)		**Lab Assignment Number: 1**
Title of Lab Assignment: Implementation of Logic programming using PROLOG Basic of Prolog DFS for the water jug solve 8- Puzzle Problem. Family Tree (objects and relations)		
DOP: 24-03-2023		**DOS: 31-03-2023**
CO: **CO1**	**PO:** **PO1, PO2, PO3, PSO1, PSO2**	**Faculty Signature:**

Student's Signature

PRACTICAL 1

AIM: Implementation of Logic programming using PROLOG Basic of Prolog DFS for the water jug solve 8- Puzzle Problem. Family Tree (objects and relations)

THEORY:

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.

Water Jug Problem:

There are two jugs of volume A litre and B litre. Neither has any measuring mark on it.There is a pump that can be used to fill the jugs with water.How can you get exactly x litre of water into the A litre jug.Assuming that we have unlimited supply of water. Let's assume we have A=4 litre and B= 3 litre jugs. And we want exactly 2 Litre water into jug A (i.e 4 litre jug) how we will do this.

Solution:

The state space for this problem can be described as the set of ordered pairs of integers (x,y)

Where,
x represents the quantity of water in the 4-gallon jug x= 0,1,2,3,4
y represents the quantity of water in 3-gallon jug y=0,1,2,3
Start State: (0,0)
Goal State: (2,0)

Generate production rules for the water jug problem

We basically perform three operations to achieve the goal.

Fill water jug.
Empty water jug
and Transfer water jug

Initialization:

Start State: (0,0)

Apply Rule 2:

Fill 3-gallon jug
Now the state is (x,3)

Iteration 1:

Current State: (x,3)

Apply Rule 7:
Pour all water from 3-gallon jug into 4-gallon jug
Now the state is (3,0)

Iteration 2:

Current State : (3,0)

Apply Rule 2:
Fill 3-gallon jug
Now the state is (3,3)

Iteration 3:

Current State:(3,3)

Apply Rule 5:
Pour water from 3-gallon jug into 4-gallon jug until 4-gallon jug is full Now the state is (4,2)

Iteration 4:

Current State : (4,2)

Apply Rule 3:
Empty 4-gallon jug
Now state is (0,2)

Iteration 5:
Current State : (0,2)

Apply Rule 9:

Pour 2 gallon water from 3 gallon jug into 4 gallon jug

Now the state is (2,0)-- Goal Achieved.

Code:

```
member(X, [X|_]).
member(X, [_|Z]) :- member(X, Z).

move(X, Y, _) :- X =:= 2, Y =:= 0, write('done'), !.
move(X, Y, Z) :- X < 4, \+member((4, Y), Z), write("fill 4 jug"), nl, move(4, Y,
[(4, Y)|Z]).
move(X, Y, Z) :- Y < 3, \+member((X, 3), Z), write("fill 3 jug"), nl, move(X, 3,
[(X, 3)|Z]).
move(X, Y, Z) :- X > 0, \+member((0, Y), Z), write("pour 4 jug"), nl, move(0,
Y, [(0, Y)|Z]).
move(X, Y, Z) :- Y > 0, \+member((X, 0), Z), write("pour 3 jug"), nl, move(X,
0, [(X, 0)|Z]).
move(X, Y, Z) :- P is X + Y, P >= 4, Y > 0, K is 4 - X, M is Y - K,
\+member((4, M), Z), write("pour from 3 jug to 4 jug"), nl, move(4, M, [(4,
M)|Z]).
move(X, Y, Z) :- P is X + Y, P >= 3, X > 0, K is 3 - Y, M is X - K,
\+member((M, 3), Z), write("pour from 4 jug to 3 jug"), nl, move(M, 3, [(M,
3)|Z]).
move(X, Y, Z) :- K is X + Y, K < 4, Y > 0, \+member((K, 0), Z), write("pour
from 3 jug to 4 jug"), nl, move(K, 0, [(K, 0)|Z]).
move(X, Y, Z) :- K is X + Y, K < 3, X > 0, \+member((0, K), Z), write("pour
from 4 jug to 3 jug"), nl, move(0, K, [(0, K)|Z]).
```

Output:

```
fill 4 jug
fill 3 jug
pour 4 jug
pour 3 jug
pour from 3 jug to 4 jug
fill 3 jug
pour from 3 jug to 4 jug
pour 4 jug
pour 3 jug
pour from 3 jug to 4 jug
done
 true
```

Conclusion :- Successfully implemented logic programming using PROLOG. Basic of Prolog DFS for the water jug solve 8- Puzzle Problem.