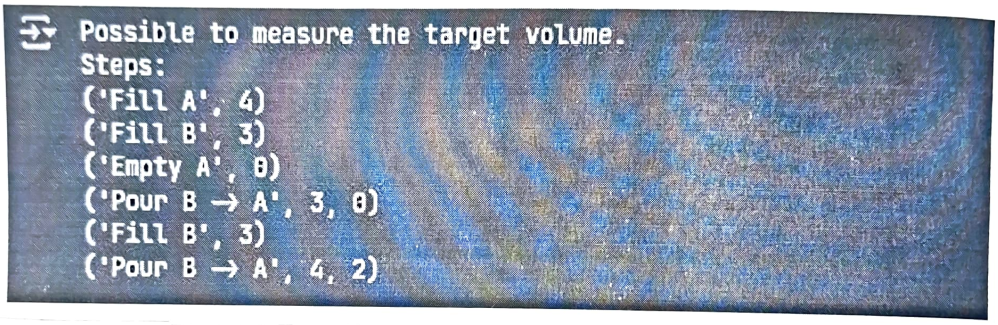
PROGRAM

```
def water_jug_dfs (capA, capB, target):
        def dfs(a, b, path):
            if (a,b) in visited:
                return false.
            visited.add((a,b))
            if a== target or b== target:
                steps. append (path)
                return True
            retur (dfs (capA, b, path + ['Fill A'])or
                dfs ( a, capB, path + [Fill B'])or
                dfs (a,b, path+ [Empty A']) or
                dfs ( a, 0, path + [empty B' ])or
                dfs (a-min (a, capB-b), b+min (a,capB-b),path +(Pour
visited=set()   dfs (a+min (b, CapA -a) , min (b,capA -a),path+ morn
Steps=[]
found=dfs(0,0,[]) return found, steps
```

GOOGLE COLLAB



```
Possible to measure the target volume.
Steps:
('Fill A', 4)
('Fill B', 3)
('Empty A', 0)
('Pour B → A', 3, 0)
('Fill B', 3)
('Pour B → A', 4, 2)
```

OUTPUT:

```
Steps:
('Fill A', 4)
('Fill B', 3)
('Empty A', 0)
('Pour B→A', 3, 0)
('Fill B'- 3)
('Pour B→A', 4, 2)
```

10/09/2024

EXPNO: 3

EXPNAME: DFS WATER JUG PROBLEM

AIM: To implement the DFS water Jug problem in python.

ALGORITHM:
1) water jug dfs(A, B, target): Main function
create a set of visited states.
2) Create sequence of operations.
3) if (a,b) has been visited return False.
4) Generate possible moves.
- Fill Jug A
- Fill Jug B
- Empty Jug A
- Empty Jug B
- Pour water from A to B
- Pour water from B to A

5) call DFS function.

RESULT:
Thus DFS has been implemented for water jug problem.