Ex No. 3 WATER JUG USING DFS

AIM:

To implement water jug problem using 2 jugs with 2 different capacities.

PSEUDOCODE:

```
function solveWaterJug(capacity_jug1, capacity_jug2, desired_quantity):
    stack = empty stack
    push initial state (0,0) onto stack
    while stack is not empty:
        current_state = pop from stack
        if current_state represents desired_quantity:
            return current_state
        generate next states from current_state
        push next states onto stack
    return "No solution found"
```

PROGRAM:

```
def solveWaterJug(capacity_jug1, capacity_jug2, desired_quantity):
    stack = []
    stack.append((0, 0))
while stack:
    current_state = stack.pop()
    if current_state[0] == desired_quantity or current_state[1] == desired_quantity:
        return current_state
    next_states = generateNextStates(current_state, capacity_jug1, capacity_jug2)
    stack.extend(next_states)
return "No solution found"
def generateNextStates(state, capacity_jug1, capacity_jug2):
    next_states = []
    next_states.append((capacity_jug1, state[1]))
    next_states.append((state[0], capacity_jug2))
    next_states.append((0, state[1]))
    next_states.append((state[0], 0))
    pour_amount = min(state[0], capacity_jug2 - state[1])
```

```
next_state.append((state[0] - pour_amount,
state[1] + pour_amount))
pour_amount = min(state[1], capacity_jug1
- state[0])
next_states.append((state[0] + pour_amount,
state[1] - pour_amount))
return next_states
solution = solveWaterJugProblem(4,3,2)
print("Solution", solution)
```

OUTPUT:

Solution: (4,2)