EXPERIMENT 4: WATERJUG PROBLEM

Aim: Implement an Algorithm in Python to solve the water jug problem.

CODE:

```python
from collections import deque

# Function to check if a state is valid
def is_valid_state(state, jug1_capacity, jug2_capacity):
    jug1, jug2 = state
    return 0 <= jug1 <= jug1_capacity and 0 <= jug2 <= jug2_capacity
def get_next_states(state, jug1_capacity, jug2_capacity):
    jug1, jug2 = state
    next_states = []

    next_states.append((jug1_capacity, jug2))
    # Fill jug2
    next_states.append((jug1, jug2_capacity))
    # Empty jug1
    next_states.append((0, jug2))
    # Empty jug2
    next_states.append((jug1, 0))
    # Pour jug1 to jug2
    pour_amount = min(jug1, jug2_capacity - jug2)
    next_states.append((jug1 - pour_amount, jug2 + pour_amount))
    # Pour jug2 to jug1
    pour_amount = min(jug2, jug1_capacity - jug1)
    next_states.append((jug1 + pour_amount, jug2 - pour_amount))

    return [state for state in next_states if is_valid_state(state, jug1_capacity,
jug2_capacity)]

# Function to solve the Water Jug Problem using BFS
def solve_water_jug(jug1_capacity, jug2_capacity, target):
    visited = set()
    queue = deque([(0, 0, [])])  # Initial state: (jug1_current, jug2_current, path)

    while queue:
        jug1_current, jug2_current, path = queue.popleft()
        state = (jug1_current, jug2_current)

        if state in visited:
            continue

        visited.add(state)
        path.append(state)

        if jug1_current == target or jug2_current == target:
            return path

        next_states = get_next_states(state, jug1_capacity, jug2_capacity)
        for next_state in next_states:
            queue.append((next_state[0], next_state[1], path.copy()))

    return None

# Function to print the solution path
def print_solution_path(path):
    if path:
        print("Solution Path:")
        for state in path:
            print(f"Jug 1: {state[0]}    Jug 2: {state[1]}")
    else:
        print("No solution exists.")



jug1_capacity = 4
jug2_capacity = 3
target = 2

solution_path = solve_water_jug(jug1_capacity, jug2_capacity, target)
print_solution_path(solution_path)
```

OUTPUT:

```
Jug 1: 0   Jug 2: 0

Jug 1: 0   Jug 2: 3

Jug 1: 3   Jug 2: 0

Jug 1: 3   Jug 2: 3

Jug 1: 4   Jug 2: 2
```

RESULT: Code has been Implemented successfully.