**Write the python program for Water Jug Problem**

**PROGRAM:**

**from collections import deque**

**def is\_goal(jug1, jug2):**

**return jug1 == 2 and jug2 == 0 # Specific goal**

**def print\_steps(path):**

**print("✅ Solution Found:")**

**for step in path:**

**print(f"Jug1: {step[0]}L, Jug2: {step[1]}L")**

**def water\_jug\_problem(jug1\_capacity, jug2\_capacity):**

**visited = set()**

**queue = deque()**

**# Start with both jugs empty**

**queue.append((0, 0, []))**

**while queue:**

**jug1, jug2, path = queue.popleft()**

**if (jug1, jug2) in visited:**

**continue**

**visited.add((jug1, jug2))**

**path = path + [(jug1, jug2)]**

**# Check if target state is reached**

**if is\_goal(jug1, jug2):**

**print\_steps(path)**

**return**

**# Generate all possible next moves**

**next\_states = []**

**# Fill Jug1**

**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 -> Jug2**

**pour\_to\_jug2 = min(jug1, jug2\_capacity - jug2)**

**next\_states.append((jug1 - pour\_to\_jug2, jug2 + pour\_to\_jug2))**

**# Pour Jug2 -> Jug1**

**pour\_to\_jug1 = min(jug2, jug1\_capacity - jug1)**

**next\_states.append((jug1 + pour\_to\_jug1, jug2 - pour\_to\_jug1))**

**for state in next\_states:**

**if state not in visited:**

**queue.append((state[0], state[1], path))**

**print("❌ No solution found.")**

**# Define capacities**

**jug1\_capacity = 4**

**jug2\_capacity = 3**

**# Run the modified problem**

**water\_jug\_problem(jug1\_capacity, jug2\_capacity)**

