**Name Aoun Muhammad**

**Roll no 045**

**Task Water jug problem**

* Problems Definition of jug / :
* **Two Jugs:** You have two jug with capacitie X and Y liter, respectively.
* **Water Source:** There's an unlimited supply of water.
* **Target Volume:** You need to measure exactly Z liter of water.
* **Operations:** You can perform the following actions:
  + Fill either jug to its capacity.
  + Empty either jug.
  + Pour water from one jug to the other until the receive jug is full or the heavy jug is empty
* **State Space:**

To each state is characterized as a pair (a, b), where (a) is the amount of water in the first jug and b is the amount in the other jug.

* **Initial State:**

The initial state is (0 , 0)  (both jug empty).

* **Goal State:**

The goal state is (z , n) or (z , n), where n is any amount of water in other jug.

***Code file:***

print("Water Jug Problem")

def goal(state, target):

    return target in state

def next\_states(state, capacities):

    next\_states = []

    x, y = state

    a, b = capacities

    next\_states.append((a, y))

    next\_states.append((x, b))

    next\_states.append((0, y))

    next\_states.append((x, 0))

    pour = min(x, b - y)

    next\_states.append((x - pour, y + pour))

    pour = min(y, a - x)

    next\_states.append((x + pour, y - pour))

    return next\_states

def dfs(start, capacities, target):

    stack = [start]

    visited = set()

    parent = {start: None}

    while stack:

        state = stack.pop()

        if goal(state, target):

            path = []

            while state:

                path.append(state)

                state = parent[state]

            path.reverse()

            return path

        if state in visited:

            continue

        visited.add(state)

        for next\_state in next\_states(state, capacities):

            if next\_state not in visited and next\_state not in parent:

                parent[next\_state] = state

                stack.append(next\_state)

    return None

def print\_solution(path):

    if path:

        print("Solution Found:")

        for step in path:

            print(f"Jug 1: {step[0]}L, Jug 2: {step[1]}L")

    else:

        print("No solution water jug problem")

jug1\_capacity = 5

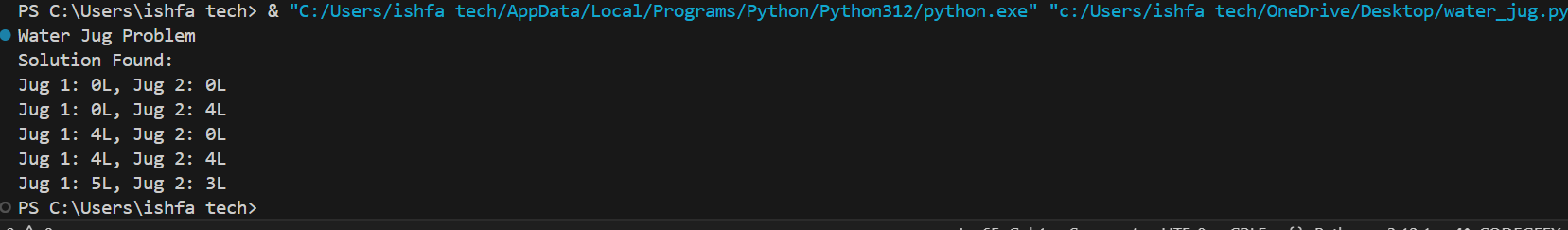
jug2\_capacity = 4

target\_amount = 3

solution = dfs((0, 0), (jug1\_capacity, jug2\_capacity), target\_amount)

print\_solution(solution )

***Output***

******