In [1]:

```
def waterjugproblem(cap1, cap2, target):
    # Stack to store states (j1, j2) and sequences of actions taken
    stack = [((0, 0), [])]  # Starts with empty jugs
    visited = set()  # To track visited states

    # List of possible actions
    actions = [
        'Fill Jug 1',
        'Fill Jug 2',
        'Empty Jug 1',
        'Empty Jug 2',
        'Pour Jug 1 -> Jug 2',
        'Pour Jug 2 -> Jug 1'
    ]

    while stack:
        (j1, j2), seq = stack.pop()

        # If state has not been visited yet
        if (j1, j2) not in visited:
            visited.add((j1, j2))

            # Check if either jug contains the target
            if j1 == target or j2 == target:
                return seq, (j1, j2)  # Return the sequence and the state

            # Generate all possible next states
            # Fill Jug 1
            if j1 < cap1:
                stack.append(((cap1, j2), seq + [actions[0]]))

            # Fill Jug 2
            if j2 < cap2:
                stack.append(((j1, cap2), seq + [actions[1]]))

            # Empty Jug 1
            if j1 > 0:
                stack.append(((0, j2), seq + [actions[2]]))

            # Empty Jug 2
            if j2 > 0:
                stack.append(((j1, 0), seq + [actions[3]]))

            # Pour Jug 1 to Jug 2
            if j1 > 0 and j2 < cap2:
                transfer = min(j1, cap2 - j2)
                stack.append(((j1 - transfer, j2 + transfer), seq + [actions[4]]))

            # Pour Jug 2 to Jug 1
            if j2 > 0 and j1 < cap1:
                transfer = min(j2, cap1 - j1)
                stack.append(((j1 + transfer, j2 - transfer), seq + [actions[5]]))

    return None

# Input and execution
cap1 = int(input("Enter the capacity of Jug 1: "))
```

```
cap2 = int(input("Enter the capacity of Jug 2: "))
target = int(input("Enter the target amount required: "))

result = waterjugproblem(cap1, cap2, target)

if result:
    seq, finalstate = result
    print("Solution found! The sequence of actions is:")
    j1, j2 = 0, 0
    for action in seq:
        if action == 'Fill Jug 1':
            j1 = cap1
        elif action == 'Fill Jug 2':
            j2 = cap2
        elif action == 'Empty Jug 1':
            j1 = 0
        elif action == 'Empty Jug 2':
            j2 = 0
        elif action == 'Pour Jug 1 -> Jug 2':
            transfer = min(j1, cap2 - j2)
            j1 -= transfer
            j2 += transfer
        elif action == 'Pour Jug 2 -> Jug 1':
            transfer = min(j2, cap1 - j1)
            j2 -= transfer
            j1 += transfer

        print(f"{action}: Jug 1 = {j1}, Jug 2 = {j2}")

    print(f"Final state: Jug 1 = {finalstate[0]}, Jug 2 = {finalstate[1]}")
else:
    print("No solution found.")
```

```
Solution found! The sequence of actions is:
Fill Jug 2: Jug 1 = 0, Jug 2 = 5
Pour Jug 2 -> Jug 1: Jug 1 = 3, Jug 2 = 2
Empty Jug 2: Jug 1 = 3, Jug 2 = 0
Pour Jug 1 -> Jug 2: Jug 1 = 0, Jug 2 = 3
Fill Jug 1: Jug 1 = 3, Jug 2 = 3
Pour Jug 1 -> Jug 2: Jug 1 = 1, Jug 2 = 5
Empty Jug 2: Jug 1 = 1, Jug 2 = 0
Pour Jug 1 -> Jug 2: Jug 1 = 0, Jug 2 = 1
Fill Jug 1: Jug 1 = 3, Jug 2 = 1
Pour Jug 1 -> Jug 2: Jug 1 = 0, Jug 2 = 4
Final state: Jug 1 = 0, Jug 2 = 4
```

In [ ]: