**PRACTICAL 2**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Name:** | Harsh Shah | **Semester:** | VI | **Division:** | 6 |
| **Roll No.:** | 21BCP359 | **Date:** | 17-01-24 | **Batch:** | G11 |
| **Aim:** | You are given two jugs with m litres and a n litre capacity. Both the jugs are initially empty. The jugs don’t have markings to allow measuring smaller quantities. You have to use the jugs to measure d litres of water where d is less than n. | | | | |

**Program**

from collections import deque

def water\_jug\_BFS(x, y, z):

    visited = set()

    queue = deque([((0, 0), [])])

    while queue:

        (jug\_a, jug\_b), actions = queue.popleft()

        if jug\_a == z or jug\_b == z or jug\_a + jug\_b == z:

            return actions + ["Success"], True

        if (jug\_a, jug\_b) in visited:

            continue

        visited.add((jug\_a, jug\_b))

        # Fill jug A

        if jug\_a < x:

            queue.append(((x, jug\_b), actions + ["Fill A"]))

        # Fill jug B

        if jug\_b < y:

            queue.append(((jug\_a, y), actions + ["Fill B"]))

        # Empty jug A

        if jug\_a > 0:

            queue.append(((0, jug\_b), actions + ["Empty A"]))

        # Empty jug B

        if jug\_b > 0:

            queue.append(((jug\_a, 0), actions + ["Empty B"]))

        # Pour from A to B

        if jug\_a + jug\_b >= y:

            queue.append(((jug\_a - (y - jug\_b), y), actions + ["Pour A to B"]))

        else:

            queue.append(((0, jug\_a + jug\_b), actions + ["Pour A to B"]))

        # Pour from B to A

        if jug\_a + jug\_b >= x:

            queue.append(((x, jug\_b - (x - jug\_a)), actions + ["Pour B to A"]))

        else:

            queue.append(((jug\_a + jug\_b, 0), actions + ["Pour B to A"]))

    return [], False

if \_\_name\_\_ == "\_\_main\_\_":

    n = int(input("Enter jug A's capacity (n): "))

    m = int(input("Enter jug B's capacity (m): "))

    d = int(input("Enter capacity to measure (d): "))

    actions, result = water\_jug\_BFS(n, m, d)

    if result:

        print("The sequence of actions is:")

        for action in actions:

            print(action)

    else:

        print("No solution found.")

**Output**



