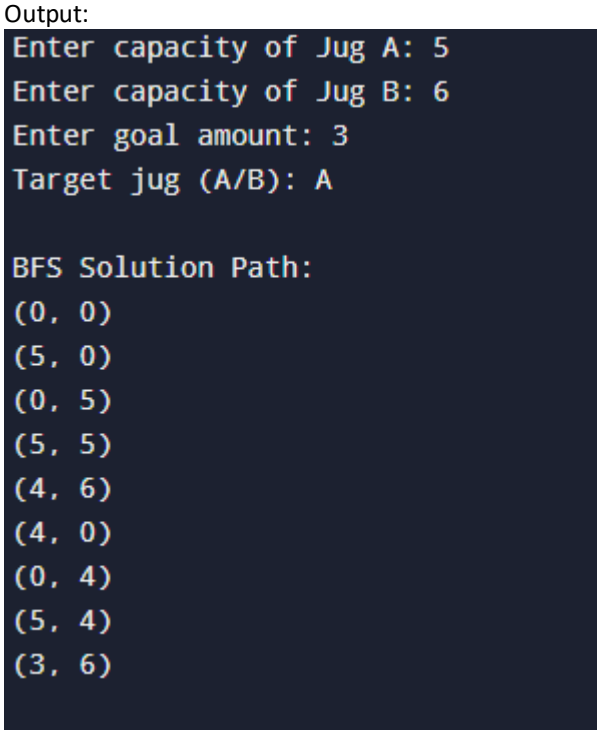
1)Waterjug model using bfs

```
from collections import deque

def water_jug_bfs(capA, capB, goal, target):
    visited = set()
    q = deque([(0, 0, [])])

    while q:
        a, b, path = q.popleft()

        if (target == 'A' and a == goal) or (target == 'B' and b == goal):
            return path + [(a, b)]

        if (a, b) in visited:
            continue
        visited.add((a, b))

        next_states = [
            (capA, b),
            (a, capB),
            (0, b),
            (a, 0),
            (a - min(a, capB - b), b + min(a, capB - b)),
            (a + min(b, capA - a), b - min(b, capA - a))
        ]

        for na, nb in next_states:
            if (na, nb) not in visited:
                q.append((na, nb, path + [(a, b)]))

    return None


capA = int(input("Enter capacity of Jug A: "))
capB = int(input("Enter capacity of Jug B: "))
goal = int(input("Enter goal amount: "))
target = input("Target jug (A/B): ").upper()

result = water_jug_bfs(capA, capB, goal, target)

if result:
    print("\nBFS Solution Path:")
    for state in result:
        print(state)
else:
    print("No solution possible")
```

Output:

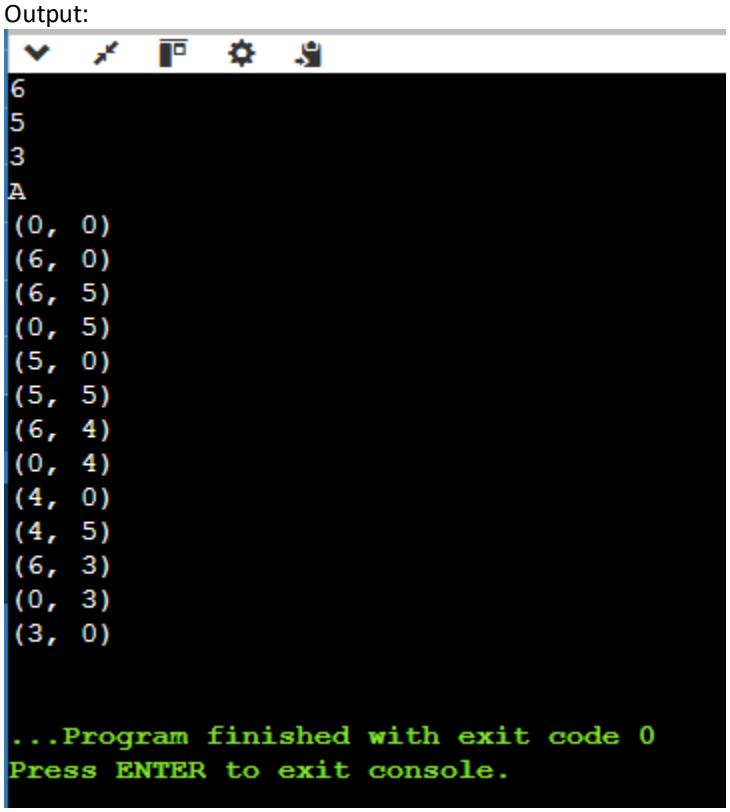


2) water jug model using dfs:

Code:

```
def dfs(a, b):
    if (a, b) in vis:
        return
    vis.add((a, b))

    if (t == 'A' and a == g) or (t == 'B' and b == g):
        print(*path + [(a, b)], sep="\n")
        exit()

    for na, nb in [(A, b), (a, B), (0, b), (a, 0),
                   (a-min(a, B-b), b+min(a, B-b)),
                   (a+min(b, A-a), b-min(b, A-a))]:
        path.append((a, b))
        dfs(na, nb)
        path.pop()


A = int(input())
B = int(input())
g = int(input())
t = input().upper()

vis, path = set(), []
dfs(0, 0)
```

Output:

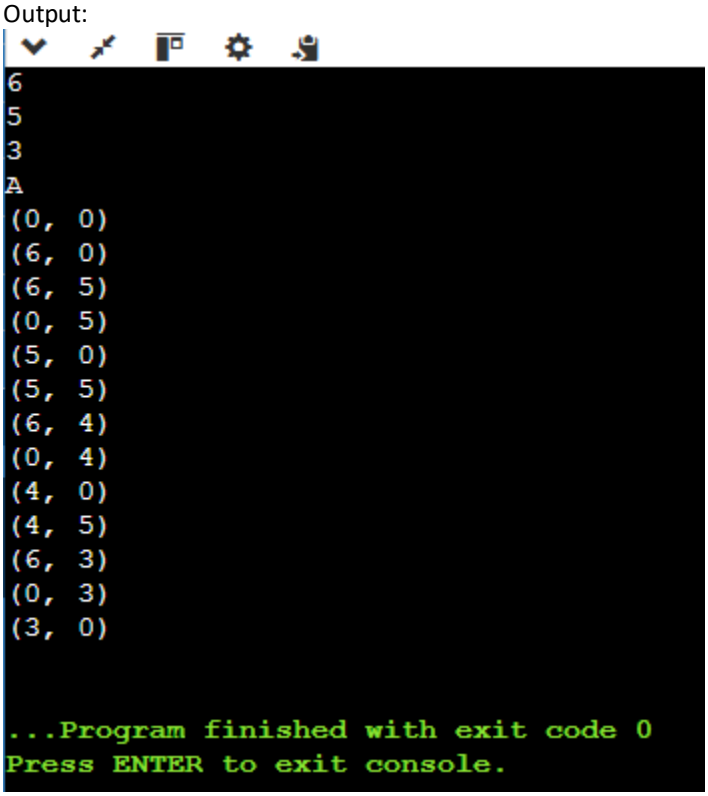


3)Water jug model using memoization:

Code:

```
def dfs(a, b):
    if (a, b) in memo:
        return
    memo.add((a, b))

    path.append((a, b))
    if (t == 'A' and a == g) or (t == 'B' and b == g):
        print(*path, sep="\n")
        exit()

    for na, nb in [(A,b),(a,B),(0,b),(a,0),
                   (a-min(a,B-b), b+min(a,B-b)),
                   (a+min(b,A-a), b-min(b,A-a))]:
        dfs(na, nb)

    path.pop()


A = int(input())
B = int(input())
g = int(input())
t = input().upper()

memo, path = set(), []
dfs(0, 0)
```

Output:


6
5
3
A
(0, 0)
(6, 0)
(6, 5)
(0, 5)
(5, 0)
(5, 5)
(6, 4)
(0, 4)
(4, 0)
(4, 5)
(6, 3)
(0, 3)
(3, 0)
...Program finished with exit code 0
Press ENTER to exit console.