PROGRAM:

```
from heapq import heappop, heappush
def heuristic(state, target, target_jug):
    if target_jug == 1:
        return abs(state[0] - target)
    else:
        return abs(state[1] - target)
def a_star(Jug1, Jug2, target, target_jug):
    open_set = []
    heappush(open_set, (0 + heuristic((0, 0), target, target_jug), 0, (0, 0), []))
    closed_set = set()
    g_cost = { (0, 0): 0 }
    while open_set:
        _, cost, current, path = heappop(open_set)
        if current in closed_set:
            continue
        path = path + [current]
        closed_set.add(current)
        if (target_jug == 1 and current[0] == target) or (target_jug == 2 and current[1] == target):
            for state in path:
                print(f"({state[0]}, {state[1]})")
            return
        for next_state in get_neighbors(current, Jug1, Jug2):
            if next_state in closed_set:
                continue
            tentative_g_cost = cost + 1
            if next_state not in g_cost or tentative_g_cost < g_cost[next_state]:
```

```
            g_cost[next_state] = tentative_g_cost
            f_cost = tentative_g_cost + heuristic(next_state, target, target_jug)
            heappush(open_set, (f_cost, tentative_g_cost, next_state, path))
    print("No solution")
def get_neighbors(state, Jug1, Jug2):
    neighbors = []
    a, b = state
    neighbors.append((Jug1, b))
    neighbors.append((a, Jug2))
    neighbors.append((0, b))
    neighbors.append((a, 0))
    transfer = min(a, Jug2 - b)
    neighbors.append((a - transfer, b + transfer))
    transfer = min(b, Jug1 - a)
    neighbors.append((a + transfer, b - transfer))
    return neighbors
Jug1 = int(input("Enter the capacity of Jug1: "))
Jug2 = int(input("Enter the capacity of Jug2: "))
target = int(input("Enter the target amount: "))
target_jug = int(input("Enter the target jug (1 for Jug1, 2 for Jug2): "))
print("Path from initial state to solution state:")
a_star(Jug1, Jug2, target, target_jug)
```

OUTPUT:

```
Enter the capacity of Jug1: 4
Enter the capacity of Jug2: 3
Enter the target amount: 2
Enter the target jug (1 for Jug1, 2 for Jug2): 1
Path from initial state to solution state:
(0, 0)
(0, 3)
(3, 0)
(3, 3)
(4, 2)
(0, 2)
(2, 0)
```