PROGRAM-3

WATER JUG PROBLEM

AIM:-

To write and execute the python program for the water jug program.

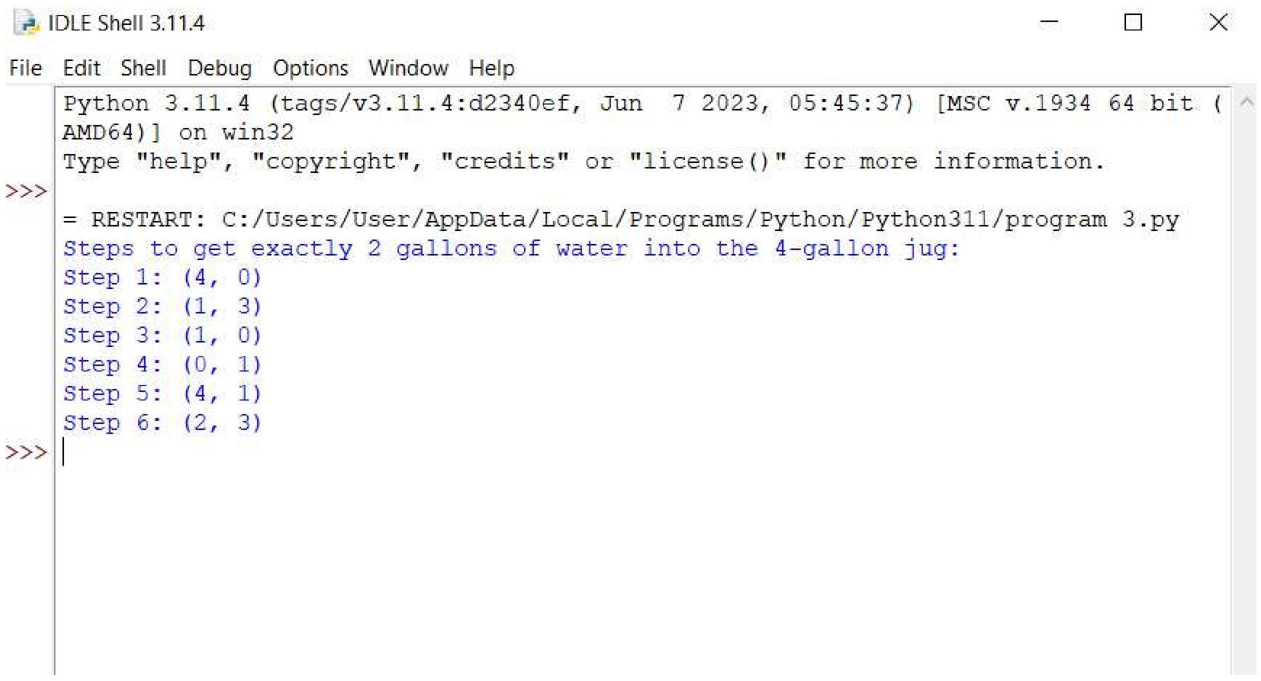
PROCEDURE:-

- **Initialize Data Structures**:
  - Create an empty set visited to keep track of visited states.
  - Create a queue queue to store the states to be explored. Initialize it with the initial state (0, 0, []).
- **Breadth-First Search (BFS)**:
  - Perform BFS traversal on the state space until a solution is found or all possible states are explored.
  - Pop a state (jug_4, jug_3, actions) from the front of the queue.
  - Check if the state represents the goal state where jug_4 contains exactly 2 gallons of water. If yes, return the sequence of actions.
  - Add the current state (jug_4, jug_3) to the set of visited states.
- **Generate Successors**:
  - Generate successor states by performing valid actions on the current state. Valid actions include:
    - Fill the 4-gallon jug.
    - Fill the 3-gallon jug.
    - Empty the 4-gallon jug.
    - Empty the 3-gallon jug.
    - Pour water from one jug to another until either jug is full or empty.
  - Add valid successor states to the queue along with the sequence of actions taken to reach them.
- **Return Solution**:
  - If a solution is found, return the sequence of actions required to achieve the goal state.
  - If no solution is found after exploring all possible states, return None

CODING:-

```
def water_jug_problem():
    visited = set()

    queue = [(0, 0, [])]

    while queue:

        jug_4, jug_3, actions = queue.pop(0)

        if jug_4 == 2:

            return actions

        visited.add((jug_4, jug_3))

        for x, y in [(4, jug_3), (jug_4, 3), (0, jug_3), (jug_4, 0), (jug_4 - min(jug_4, 3 -
jug_3), jug_3 + min(jug_4, 3 - jug_3)), (jug_4 + min(jug_3, 4 - jug_4), jug_3 - min(jug_3,
4 - jug_4))]:

            if 0 <= x <= 4 and 0 <= y <= 3 and (x, y) not in visited:

                queue.append((x, y, actions + [(x, y)]))

    return None



solution = water_jug_problem()

if solution:

    print("Steps to get exactly 2 gallons of water into the 4-gallon jug:")

    for step, action in enumerate(solution):

        print(f"Step {step + 1}: {action}")

else:

    print("No solution found.")
```

OUTPUT:-



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun  7 2023, 05:45:37) [MSC v.1934 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:/Users/User/AppData/Local/Programs/Python/Python311/program 3.py
Steps to get exactly 2 gallons of water into the 4-gallon jug:
Step 1: (4, 0)
Step 2: (1, 3)
Step 3: (1, 0)
Step 4: (0, 1)
Step 5: (4, 1)
Step 6: (2, 3)
>>>
```

RESULT:-

Hence the program has been successfully executed and verified.