Experiment No.1

Semester	T.E. Semester VI
Subject	ARTIFICIAL INTELLIGENCE (CSL 604)
Subject Professor In-charge	Prof. Avinash Shrivas
Assisting Teachers	Prof. Avinash Shrivas

Student Name	Deep Salunkhe
Roll Number	21102A0014
Lab Number	310A

Title:

Water Jug Problem (AI)

Theory:

The water jug problem is a classic puzzle that involves two jugs with known capacities, and the task is to measure a specific amount of water using these jugs. The jugs can be filled, emptied, and water can be transferred between them.

Objective:

In this program, the objective is to make the 4g jug contain exactly 2g of water.

Jug Capacities:

- 4g Jug (g_4): Represents the jug with a capacity of 4 units.
- 3g Jug (g_3): Represents the jug with a capacity of 3 units.

Program Explanation:

Initial State:

The program starts with both jugs empty (0g in both).

Program Code:

```cpp
#include <iostream>
#include <unordered_map>
#include <vector>
using namespace std;

// Define a custom hash function for pairs of integers
struct pair_hash {
    template <class T1, class T2>
    size_t operator()(const pair<T1, T2>& p) const {
        auto hash1 = hash<T1>{}(p.first);
        auto hash2 = hash<T2>{}(p.second);
        return hash1 ^ hash2;
    }
};

// Function to check if the current state is the goal state
bool isGoalState(int g_4, int g_3) {
    return g_4 == 2; // Goal state: 4g jug has 2g water, 3g jug is empty
}


// Function to solve the water jug problem
void solveWaterJugProblem() {
    // Map to store the next states for each current state
    unordered_map<pair<int, int>, pair<int, int>, pair_hash> nextStateMap;
    vector<string> path; // Path to store the solution steps
    int g_4 = 0, g_3 = 0; // Initial state

    // Generate the nextStateMap based on the rules of the problem
    nextStateMap[{0, 0}] = {0, 3};
    nextStateMap[{0, 3}] = {3, 0};
    nextStateMap[{3, 0}] = {3, 3};
    nextStateMap[{3, 3}] = {4, 2};
    nextStateMap[{0, 2}] = {2, 0};
    nextStateMap[{2, 0}] = {2, 3};
    nextStateMap[{2, 3}] = {4, 1};
    nextStateMap[{4, 1}] = {1, 0};
    nextStateMap[{1, 0}] = {1, 3};
    nextStateMap[{1, 3}] = {4, 0};
    nextStateMap[{4, 0}] = {2, 0};
    nextStateMap[{4, 2}] = {0, 2};

    // Iterative process to find the solution
```

```
    while (!isGoalState(g_4, g_3)) {
        // Print the current state
        cout << "Prev State: [" << g_4 << ", " << g_3 << "] => ";

        // Get the next state from the map
        auto nextState = nextStateMap[{g_4, g_3}];

        // Print the next state
        cout << "Next State: [" << nextState.first << ", " << nextState.second <<
"]" << endl;

        // Update the current state
        g_4 = nextState.first;
        g_3 = nextState.second;
    }

    // Print the final state
     cout<<"Goal reached"<<endl;
}

int main() {
    cout << "Let's start the water jug problem\n";
    cout << "Task: make 4g jug have 2g water\n";

    // Solve the water jug problem
    solveWaterJugProblem();

    return 0;
}
```

Output:

```
Let's start the water jug problem
Task: make 4g jug have 2g water
Prev State: [0, 0] => Next State: [0, 3]
Prev State: [0, 3] => Next State: [3, 0]
Prev State: [3, 0] => Next State: [3, 3]
Prev State: [3, 3] => Next State: [4, 2]
Prev State: [4, 2] => Next State: [0, 2]
Prev State: [0, 2] => Next State: [2, 0]
Goal reached
PS E:\GIt\SEM-6\AI>
```

Conclusion:

The water jug problem is a classic example of problem-solving using basic operations such as filling, emptying, and transferring. The provided C++ program solves the problem and using the state mapping logic