Experiment-3

Student Name: Tejinderpal Singh **UID:** 23BCS80343
Branch: BE-CSE **Section/Group:** 23BCS_KRG_3A
Semester:5th **Subject Code:** 23CSH-301
Subject Name: DAA

1. **Aim:** Evaluate the complexity of the developed program to find frequency of elements in a given array.

2. **Objective:** To write a C++ program that finds the frequency of each element in a given array using an efficient method called hashing. The program uses an unordered_map to store and count occurrences of elements, allowing it to run in O(n) time instead of the slower O(n²) approach. The program processes a fixed array, calculates how often each element appears, and displays the result.

3. **Procedure:**

   1. Start.
   2. Declare and initialize an integer array.
   3. Calculate the size of the array using the formula:
      n = sizeof(array) / sizeof(array[0])
   4. Create an empty hash map to store:
      - Keys → unique elements of the array
      - Values → corresponding frequency of each element
   5. Traverse the array from index 0 to n - 1:
      - For each element, increase its count in the hash map.
   6. Iterate through the hash map:
      - For each key-value pair, print the element and its frequency.
   7. End.

4. **Code:**
```
#include<iostream>
#include <unordered_map>
using namespace std;
void findFrequency(int arr[], int n) {
    unordered_map<int, int> freq;

    for (int i = 0; i < n; i++) {
        freq[arr[i]]++;
```
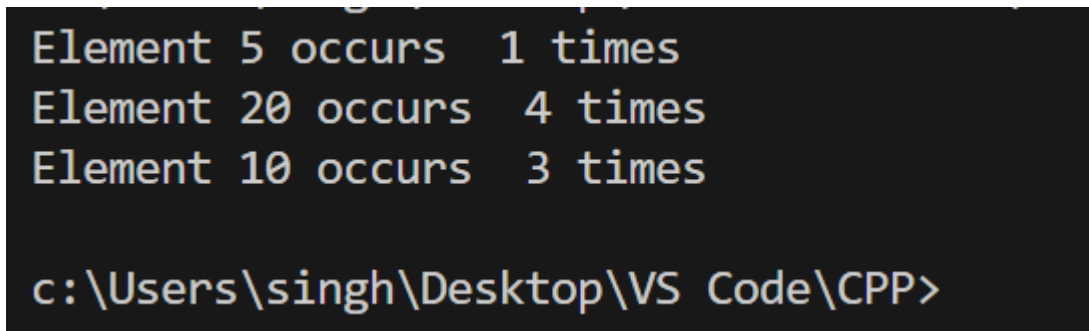
```
    }

    for (auto x : freq) {
        cout << "Element " << x.first << " occurs " << " " << x.second << " times" <<
endl;
    }
}
int main(){
    int arr[]={10, 20, 20, 10, 10, 20, 5, 20};
    int n=sizeof(arr)/sizeof(arr[0]);
    findFrequency(arr, n);
    return 0;
}
```

5. Observations:

```
Element 5 occurs  1 times
Element 20 occurs  4 times
Element 10 occurs  3 times

c:\Users\singh\Desktop\VS Code\CPP>
```

6. Time Complexity: O(n)

7. Learning Outcome:

❖ Learned how to use unordered_map in C++ to count the frequency of array elements efficiently.
❖ Understood how hashing helps achieve O(n) time complexity for frequency counting.
❖ Practiced using loops and STL containers to process and display data.
❖ Gained experience in modular programming by separating logic into a dedicated function.
❖ Strengthened skills in writing clean, optimized, and readable C++ code.