**Assignment-51** **A Job Ready Bootcamp in c++,DSA and IOT**

**unordered_multiset**

1. Write a C++ program to initialize the unordered_multiset and print it on the screen.

```cpp
#include <bits/stdc++.h>
using namespace std;

// making typedef for short declaration
typedef unordered_multiset<int>::iterator umit;

// Utility function to print unordered_multiset
void printUset(unordered_multiset<int> ums)
{
    // begin() returns iterator to first element of set
    umit it = ums.begin();
    for (; it != ums.end(); it++)
        cout << *it << " ";
    cout << endl;
}

// Driver program to check all function
int main()
{
    // empty initialization
    unordered_multiset<int> ums1;

    // Initialization by intializer list
    unordered_multiset<int> ums2 ({1, 3, 1, 7, 2, 3, 4, 1, 6});

    // Initialization by assignment
    ums1 = {2, 7, 2, 5, 0, 3, 7, 5};

    // empty() function return true if set is empty
    // otherwise false
    if (ums1.empty())
        cout << "unordered multiset 1 is empty\n";
    else
        cout << "unordered multiset 1 is not empty\n";

    // size() function returns total number of elements
    // in data structure
    cout << "The size of unordered multiset 2 is : " << ums2.size() << endl;

    printUset(ums1);

    ums1.insert(7);

    printUset(ums1);

    int val = 3;

    // find function returns iterator to first position
    // of val, if exist otherwise it returns iterator
    // to end
    if (ums1.find(val) != ums1.end())
        cout << "unordered multiset 1 contains " << val << endl;
    else
        cout << "unordered multiset 1 does not contains " << val << endl;

    // count function returns total number of occurrence in set
    val = 5;
    int cnt = ums1.count(val);
    cout << val << " appears " << cnt << " times in unordered multiset 1 \n";
```

```
    val = 9;

    // if count return >0 value then element exist otherwise not
    if (ums1.count(val))
        cout << "unordered multiset 1 contains "
            << val << endl;
    else
        cout << "unordered multiset 1 does not contains "
            << val << endl;

    val = 1;

    // equal_range returns a pair, where first is iterator
    // to first position of val and second it iterator to
    // last position to val
    pair<umit, umit> erange_it = ums2.equal_range(val);
    if (erange_it.first != erange_it.second)
        cout << val << " appeared atleast once in unoredered_multiset \n";


    printUset(ums2);

    // erase function deletes all instances of val
    ums2.erase(val);

    printUset(ums2);

    // clear function deletes all entries from set
    ums1.clear();
    ums2.clear();

    if (ums1.empty())
        cout << "unordered multiset 1 is empty\n";
    else
        cout << "unordered multiset 1 is not empty\n";
}
```

```
Output:
unordered multiset 1 is not empty
The size of unordered multiset 2 is : 9
3 0 5 5 7 7 2 2
3 0 5 5 7 7 7 2 2
unordered multiset 1 contains 3
5 appears 2 times in unordered multiset 1
unordered multiset 1 does not contains 9
1 appeared atleast once in unoredered_multiset
6 4 2 7 3 3 1 1 1
6 4 2 7 3 3
unordered multiset 1 is empty
```

2. Write a C++ program to delete all copies from an unordered_multiset.
   Example:
   Input - 6 4 2 7 3 3 1 1 1
   Output - 6 4 2 7 3 1

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    unordered_multiset<int> ums;
    unordered_multiset<int>::iterator it;
```

```
    // Initialization by assignment
    ums = {6, 4, 2, 7, 3, 3, 1, 1, 1};

    for ( it = ums.begin(); it != ums.end(); it++)
        cout << *it << " ";
    auto val = ums.find(1);
    ums.erase(val);
    auto val1 = ums.find(3);
    ums.erase(val1);
    cout << endl;

    for ( it = ums.begin(); it != ums.end(); it++)
        cout << *it << " ";

    return 0;
}
```

```
Output:
1 1 1 3 3 7 2 4 6
1 1 3 7 2 4 6
```

3. Given an array arr[] of N integer elements, the task is to change the minimum number of elements of this array such that it contains first N terms of the Catalan Sequence. Thus, find the minimum changes required using unordered_multiset.
   First few Catalan numbers are 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, .....
   Examples:
   Input: arr[] = {4, 1, 2, 33, 213, 5}
   Output: 3
   We have to replace 4, 33, 213 with 1, 14, 42 to make the first 6 terms of Catalan sequence.
   Input: arr[] = {1, 1, 2, 5, 41}
   Output: 1
   Simply change 41 with 14

```
#include <bits/stdc++.h>
using namespace std;
#define MAX 100000
#define ll long long int

// To store first N Catalan numbers
ll catalan[MAX];

// Function to find first n Catalan numbers
void catalanDP(ll n)
{

    // Initialize first two values in table
    catalan[0] = catalan[1] = 1;

    // Fill entries in catalan[] using recursive formula
    for (int i = 2; i <= n; i++) {
        catalan[i] = 0;
        for (int j = 0; j < i; j++)
            catalan[i] += catalan[j] * catalan[i - j - 1];
    }
}

// Function to return the minimum changes required
int CatalanSequence(int arr[], int n)
{
```

```
    // Find first n Catalan Numbers
    catalanDP(n);

    unordered_multiset<int> s;

    // a and b are first two
    // Catalan Sequence numbers
    int a = 1, b = 1;
    int c;

    // Insert first n catalan elements to set
    s.insert(a);
    if (n >= 2)
        s.insert(b);

    for (int i = 2; i < n; i++)
        s.insert(catalan[i]);

    unordered_multiset<int>::iterator it;

    for (int i = 0; i < n; i++) {

        // If catalan element is present
        // in the array then remove it from set
        it = s.find(arr[i]);
        if (it != s.end())
            s.erase(it);
    }

    // Return the remaining number of
    // elements in the set
    return s.size();
}

// Driver code
int main()
{
    int arr[] = { 4, 1, 2, 33, 213, 5 };
    int n = sizeof(arr) / sizeof(arr[0]);

    cout << CatalanSequence(arr, n);

    return 0;
}
================================================================================
Output:
3
```

4. Write a C++ program to illustrate the swapping of data between two unordered_multiset.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    unordered_multiset<int> ums1;
    unordered_multiset<int> ums2;
    unordered_multiset<int>::iterator it;

    // Initialization by assignment
    ums1 = {6, 4, 2, 7, 3, 3, 1, 1, 1};
    ums2 = {8, 2, 7, 4, 45, 69, 27, 33};

    cout << "Before swapping" << endl;
```

```
    for ( it = ums1.begin(); it != ums1.end(); it++)
        cout << *it << " ";
    cout << endl;
    for ( it = ums2.begin(); it != ums2.end(); it++)
        cout << *it << " ";

    ums1.swap(ums2);

    cout << "\nAfter swapping" << endl;
    for ( it = ums1.begin(); it != ums1.end(); it++)
        cout << *it << " ";
    cout << endl;
    for ( it = ums2.begin(); it != ums2.end(); it++)
        cout << *it << " ";


    return 0;
}
```

```
Output:
Before swapping
1 1 1 3 3 7 2 4 6
33 27 69 45 4 7 2 8
After swapping
33 27 69 45 4 7 2 8
1 1 1 3 3 7 2 4 6
```

5. Write a C++ program to count the frequency of elements in unordered_multiset.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    unordered_multiset<int> ums;
    unordered_multiset<int>::iterator it;
    int no;
    // Initialization by assignment
    ums = {6, 4, 2, 7, 3, 3, 1, 1, 1};
    for ( it = ums.begin(); it != ums.end(); it++)
        cout << *it << " ";
    cout << endl << "Enter a number from the above list: ";
    cin >> no;
    cout << "Frequency: " << count(ums.begin(), ums.end(), no) << endl;
    return 0;
}
```

```
Output:
1 1 1 3 3 7 2 4 6
Enter a number from the above list: 1
Frequency: 3
```

6. Write a C++ program to illustrate the emplace() function in unordered_multiset.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{

    // declaration
    unordered_multiset<int> sample;

    // inserts element using emplace()
    sample.emplace(11);
```

```
    sample.emplace(11);
    sample.emplace(11);
    sample.emplace(12);
    sample.emplace(13);
    sample.emplace(13);
    sample.emplace(14);

    cout << "Elements: ";

    for (auto it = sample.begin(); it != sample.end(); it++)
        cout << *it << " ";
    return 0;
}
================================================================================
Output:
Elements: 14 11 11 11 12 13 13
```

7. Write a C++ program to illustrate the find() function in unordered_multiset.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    unordered_multiset<int> ums;
    unordered_multiset<int>::iterator it;

    // Initialization by assignment
    ums = {6, 4, 2, 7, 3, 3, 1, 1, 1};

    for ( it = ums.begin(); it != ums.end(); it++)
        cout << *it << " ";
    int val;
    cout << "\nEnter a number: ";
    cin >> val;

    if (ums.find(val) != ums.end())
        cout << "Value found";
    else
        cout << "Value not found";

    return 0;
}
================================================================================
Output:
1 1 1 3 3 7 2 4 6
Enter a number: 2
Value found
```

8. Write a C++ program to illustrate the bucket_count() function in unordered_multiset.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{

    // declaration
    unordered_multiset<int> sample;

    // inserts element
    sample.insert(5);
    sample.insert(25);
    sample.insert(14);
    sample.insert(50);
```

```
    sample.insert(10);

    cout << "The total count of buckets: " << sample.bucket_count();

    // prints all element bucket wise
    for (int i = 0; i < sample.bucket_count(); i++)
    {
        cout << "\nBucket " << i << ": ";

        // if bucket is empty
        if (sample.bucket_size(i) == 0)
            cout << "empty";

        for (auto it = sample.cbegin(i); it != sample.cend(i); it++)
            cout << *it << " ";
    }

    return 0;
}
```

```
Output:
The total count of buckets: 7
Bucket 0: 14
Bucket 1: 50
Bucket 2: empty
Bucket 3: 10
Bucket 4: 25
Bucket 5: 5
Bucket 6: empty
```

9. Write a C++ program to illustrate the load_factor() function in unordered_multiset.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{

    // declaration
    unordered_multiset<int> sample;

    // inserts element
    sample.insert(1);
    sample.insert(1);

    cout << "The size is: " << sample.size();
    cout << "\nThe bucket_count is: " << sample.bucket_count();
    cout << "\nThe load_factor is: " << sample.load_factor();

    sample.insert(1);
    sample.insert(2);

    cout << "\n\nThe size is: " << sample.size();
    cout << "\nThe bucket_count is: " << sample.bucket_count();
    cout << "\nThe load_factor is: " << sample.load_factor();

    sample.insert(2);

    cout << "\n\nThe size is: " << sample.size();
    cout << "\nThe bucket_count is: " << sample.bucket_count();
    cout << "\nThe load_factor is: " << sample.load_factor();

    return 0;
}
```

```
Output:
The size is: 2
The bucket_count is: 3
The load_factor is: 0.666667

The size is: 4
The bucket_count is: 7
The load_factor is: 0.571429

The size is: 5
The bucket_count is: 7
The load_factor is: 0.714286
```

10. Write a C++ program to illustrate the reverse() function in unordered_multiset.

```cpp
#include <iostream>
#include <unordered_set>

using namespace std;

int main()
{
    unordered_multiset<int> j;

    // container invoked
    // it reverse the values
    j.reserve(5);

    // set the values of the container
    j.insert(5);
    j.insert(6);
    j.insert(7);
    j.insert(1);
    j.insert(8);

    cout << "The values in unordered_multiset :";
    for (const int& x : j)
        cout << " " << x;

    return 0;
}
```

```
================================================================================
Output:
The values in unordered_multiset : 8 1 7 6 5
```