Assignment-48 **A Job Ready Bootcamp in c++,DSA and IOT**

multimap

1. Write a c++ program to demonstrate the implementation of multimap

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    return 0;
}
================================================================================
Output:
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
```

2. Declare a multimap m1 of key-value pairs of integer type and then insert some pair type data. Now print the multimap values of m1, and also create another multimap m2 of the same type as m1 using m1.begin() and m1.end() as parameters.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    multimap<int, int>::iterator it;
    cout << "First multimap" << endl;
    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    cout << "Second multimap" << endl;
    multimap<int, int> mp2(mp.begin(), mp.end());
```

```
    multimap<int, int>::iterator itr;
    for (itr = mp.begin(); itr != mp.end(); itr++)
        cout << "Key : " << itr -> first << ", Value : " << itr -> second <<
endl;

    return 0;
}
```

```
Output:
First multimap
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
Second multimap
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
```

3. Write a c++ code for illustration of multimap::swap() function.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    multimap<int, int>::iterator it;
    cout << "First multimap" << endl;
    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    multimap<int, int> mp2;
    mp2.insert(pair<int, int> (20, 100));
    mp2.insert(pair<int, int> (10, 200));
    mp2.insert(pair<int, int> (30, 40));
    mp2.insert(pair<int, int> (40, 550));
    mp2.insert(pair<int, int> (60, 90));
    mp2.insert(pair<int, int> (45, 60));

    cout << "Second multimap" << endl;
    for (auto i = mp2.begin(); i != mp2.end(); i++)
        cout << "Key : " << i -> first << ", Value : " << i -> second <<
endl;

    mp.swap(mp2);
    cout << "After Swap" << endl;
    cout << "First multimap" << endl;
    for (it = mp.begin(); it != mp.end(); it++)
```

```
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;
    cout << "Second multimap" << endl;
    for (auto i = mp2.begin(); i != mp2.end(); i++)
        cout << "Key : " << i -> first << ", Value : " << i -> second <<
endl;
    return 0;
}
```

```
================================================================================
Output:
First multimap
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
Second multimap
Key : 10, Value : 200
Key : 20, Value : 100
Key : 30, Value : 40
Key : 40, Value : 550
Key : 45, Value : 60
Key : 60, Value : 90
After Swap
First multimap
Key : 10, Value : 200
Key : 20, Value : 100
Key : 30, Value : 40
Key : 40, Value : 550
Key : 45, Value : 60
Key : 60, Value : 90
Second multimap
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
```

4. Write a program to erase all the entries of the key.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    cout << "Size of multimap : " << mp.size() << endl;

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;
```

```
    mp.erase(mp.begin(), mp.end());

    if(mp.empty())
        cout << "Multimap is empty" << endl;

    cout << "Size of multimap : " << mp.size() << endl;
    return 0;
}
```

```
Output:
Size of multimap : 6
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
Multimap is empty
Size of multimap : 0
```

5. Write a program to erase only a single value based on position.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    cout << "Size of multimap : " << mp.size() << endl;

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    mp.erase(4);

    cout << "After erase" << endl;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    if(mp.empty())
        cout << "Multimap is empty" << endl;

    cout << "Size of multimap : " << mp.size() << endl;
    return 0;
}
```

```
Output:
Size of multimap : 6
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
```

```
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
After erase
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 5, Value : 60
Size of multimap : 4
```

6. Write a program to find some key value pairs and print on the console.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    int n;
    cout << "Enter key to find : ";
    cin >> n;
    auto i = mp.find(n);
    cout << "Key : " << i -> first << ", Value : " << i -> second << endl;

    return 0;
}
```

```
================================================================================
Output:
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
Enter key to find : 3
Key : 3, Value : 30
```

7. Write a program to find a lower bound.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (9, 10));
    mp.insert(pair<int, int> (8, 20));
    mp.insert(pair<int, int> (5, 30));
```

```
    mp.insert(pair<int, int> (2, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (12, 60));

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    int n;
    cout << "Enter number for lower bound : ";
    cin >> n;
    auto i = mp.lower_bound(n);
    cout << "Key : " << i -> first << ", Value : " << i -> second << endl;

    return 0;
}
================================================================================
Output:
Key : 2, Value : 40
Key : 4, Value : 50
Key : 5, Value : 30
Key : 8, Value : 20
Key : 9, Value : 10
Key : 12, Value : 60
Enter number for lower bound : 6
Key : 8, Value : 20
```

8. Write a program to find the upper bound.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    mp.insert(pair<int, int> (9, 10));
    mp.insert(pair<int, int> (8, 20));
    mp.insert(pair<int, int> (5, 30));
    mp.insert(pair<int, int> (2, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (12, 60));

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    int n;
    cout << "Enter number for lower bound : ";
    cin >> n;
    auto i = mp.upper_bound(n);
    cout << "Key : " << i -> first << ", Value : " << i -> second << endl;

    return 0;
}
================================================================================
Output:
Key : 2, Value : 40
Key : 4, Value : 50
Key : 5, Value : 30
```

```
Key : 8, Value : 20
Key : 9, Value : 10
Key : 12, Value : 60
Enter number for lower bound : 8
Key : 9, Value : 10
```

9. You are given an array A of size N. You need to insert the elements of A into a multimap(element as key and index as value) and display the results. Also, you need to erase a given element x from the multimap and print "erased x" if successfully erased, else print "not found".

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    int arr[] = {10, 21, 45 ,87, 24, 63, 89, 54, 78, 96};
    multimap<int, int> mp;
    for (int i = 0; i < 10; i++)
        mp.insert(pair<int, int> (arr[i], i));

    multimap<int, int>::iterator it, it1;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;

    int n;
    cout << "Enter key you want to erase : ";
    cin >> n;
    it1 = mp.find(n);

    if (it1 == mp.end())
    {
        cout << "Not found" << endl;
    }
    else
    {
        cout << "Key : " << it1 -> first << ", Value : " << it1 -> second <<
" is successfully erased" << endl << endl;
        mp.erase(n);
        for (it = mp.begin(); it != mp.end(); it++)
            cout << "Key : " << it -> first << ", Value : " << it -> second
<< endl;
    }

    return 0;
}
```

```
Output:
Key : 10, Value : 0
Key : 21, Value : 1
Key : 24, Value : 4
Key : 45, Value : 2
Key : 54, Value : 7
Key : 63, Value : 5
Key : 78, Value : 8
Key : 87, Value : 3
Key : 89, Value : 6
Key : 96, Value : 9
Enter key you want to erase : 89
Key : 89, Value : 6 is successfully erased
```

```
Key : 10, Value : 0
Key : 21, Value : 1
Key : 24, Value : 4
Key : 45, Value : 2
Key : 54, Value : 7
Key : 63, Value : 5
Key : 78, Value : 8
Key : 87, Value : 3
Key : 96, Value : 9
```

10. Write a program that checks whether a given multimap is empty or not.

```
#include <iostream>
#include <map>
#include <iterator>
using namespace std;

int main()
{
    multimap<int, int> mp;
    if (mp.empty())
        cout << "Multimap is empty" << endl;
    else
        cout << "Multimap is not empty" << endl;

    cout << "Size of multimap : " << mp.size() << endl;
    mp.insert(pair<int, int> (1, 10));
    mp.insert(pair<int, int> (2, 20));
    mp.insert(pair<int, int> (3, 30));
    mp.insert(pair<int, int> (4, 40));
    mp.insert(pair<int, int> (4, 50));
    mp.insert(pair<int, int> (5, 60));

    multimap<int, int>::iterator it;

    for (it = mp.begin(); it != mp.end(); it++)
        cout << "Key : " << it -> first << ", Value : " << it -> second <<
endl;
    cout << "Size of multimap : " << mp.size() << endl;
    if (mp.empty())
        cout << "Multimap is empty" << endl;
    else
        cout << "Multimap is not empty" << endl;

    return 0;
}
```

```
================================================================================
Output:
Multimap is empty
Size of multimap : 0
Key : 1, Value : 10
Key : 2, Value : 20
Key : 3, Value : 30
Key : 4, Value : 40
Key : 4, Value : 50
Key : 5, Value : 60
Size of multimap : 6
Multimap is not empty
```