multiset

1. Create a c++ program using multiset and returns an iterator to the first element in the multiset –> O(1)

```cpp
#include <iostream>
#include <set>
#include <iterator>
using namespace std;

int main()
{
    multiset<int, greater<int>> ms1;
    ms1.insert(10);
    ms1.insert(20);
    ms1.insert(30);
    ms1.insert(40);
    ms1.insert(50);
    ms1.insert(40);

    multiset<int, greater<int>>:: iterator it1, it2;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    cout << "\nFirst element: " << *ms1.begin() << endl;
    return 0;
}
```

```
================================================================================
Output:
50 40 40 30 20 10
First element: 50
```

2. Create a c++ program using multiset and returns an iterator to the theoretical element that follows the last element in the multiset –> O(1)

```cpp
#include <iostream>
#include <set>
#include <iterator>
using namespace std;
int main()
{

    int arr[] = { 14, 10, 15, 11, 20 };

    multiset<int, greater<int>> ms1;
    multiset<int, greater<int>>:: iterator it1;

    for(int i = 0; i < 5; i++)
        ms1.insert(arr[i]);

    it1 = ms1.end();
    it1--;

    // Print the last element
    cout << "The last element is: " << *it1 << endl;

    // prints all elements in set
    for (auto it = ms1.begin(); it != ms1.end(); it++)
        cout << *it << " ";

    return 0;
}
```

```
================================================================================
Output:
The last element is: 10
20 15 14 11 10
```

3. Create a c++ program using multiset and returns the number of elements in the multiset –> O(1)

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

int main()
{
    multiset<int, greater<int>> ms1;
    ms1.insert(10);
    ms1.insert(20);
    ms1.insert(30);
    ms1.insert(40);
    ms1.insert(50);
    ms1.insert(60);

    multiset<int, greater<int>>:: iterator it1;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    cout << "\nNumber of elenents: " << ms1.size() << endl;

    return 0;
}
================================================================================
Output:
60 50 40 30 20 10
Number of elenents: 6
```

4. Create a c++ program using multiset and returns the maximum number of elements that the multiset can hold –> O(1)

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

int main()
{
    multiset<int, greater<int>> ms1;
    ms1.insert(10);
    ms1.insert(20);
    ms1.insert(30);
    ms1.insert(40);
    ms1.insert(50);
    ms1.insert(60);

    multiset<int, greater<int>>:: iterator it1;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    cout << "\nNumber of elenents: " << ms1.size() << endl;
     cout << "The maximum number of elements that the multiset can hold: " <<
ms1.max_size() << endl;

    return 0;
}
```

```
================================================================================
Output:
60 50 40 30 20 10
Number of elenents: 6
The maximum number of elements that the multiset can hold: 230584300921369395
```

5. Create a c++ program using multiset and returns whether the multiset is empty → O(1)

```cpp
#include <iostream>
#include <set>
#include <iterator>
using namespace std;

int main()
{
    multiset<int, greater<int>> ms1;

    if (ms1.empty())
        cout << "multiset is empty" << endl;
    else
        cout << "multiset not is empty" << endl;

    ms1.insert(10);
    ms1.insert(20);
    ms1.insert(30);
    ms1.insert(40);
    ms1.insert(50);
    ms1.insert(60);

    multiset<int, greater<int>>:: iterator it1, it;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    if (ms1.empty())
        cout << "\nmultiset is empty" << endl;
    else
        cout << "\nmultiset not is empty" << endl;


    return 0;
}
```

```
================================================================================
Output:
multiset is empty
60 50 40 30 20 10
multiset not is empty
```

6. Create a c++ program using multiset and inserts the element x in the multiset → O(log n)

```cpp
#include <iostream>
#include <set>
#include <iterator>
using namespace std;

int main()
{
    multiset<int, greater<int>> ms1;
    int number, x;
    cout << "Enter number of elements to store in multiset: ";
    cin >> number;
    cout << "Enter " << number << " numbers: ";
    for(int i = 0; i < number; i++)
    {
        cin >> x;
```

```
        ms1.insert(x);
    }

    multiset<int, greater<int>>:: iterator it1;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    return 0;
}
```

```
Output:
Enter number of elements to store in multiset: 6
Enter 6 numbers: 6 8 9 4 3 1
9 8 6 4 3 1
```

7. Create a c++ program using multiset and removes all the elements from the multiset –> O(n)

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

int main()
{
    multiset<int, greater<int>> ms1;
    int number, x;
    cout << "Enter number of elements to store in multiset: ";
    cin >> number;
    cout << "Enter " << number << " numbers: ";
    for(int i = 0; i < number; i++)
    {
        cin >> x;
        ms1.insert(x);
    }

    multiset<int, greater<int>>:: iterator it1;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    ms1.clear();
      cout << "\nAfter removing all elements, Size of multiset is   " <<
ms1.size() << endl;
    if(ms1.empty())
        cout <<"multiset is empty" << endl;
    else
        cout <<"multiset is not empty" << endl;

    return 0;
}
```

```
Output:
Enter number of elements to store in multiset: 7
Enter 7 numbers: 54 23 14 56 89 85 74
89 85 74 56 54 23 14
After removing all elements, Size of multiset is  0
multiset is empty
```

8. Create a c++ program using multiset and removes all the occurrences of x –> O(log n)

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

```
int main()
{
    multiset<int> ms1;
    int number, x;
    cout << "Enter number of elements to store in multiset: ";
    cin >> number;
    cout << "Enter " << number << " numbers: ";
    for(int i = 0; i < number; i++)
    {
        cin >> x;
        ms1.insert(x);
    }

    multiset<int, greater<int>>:: iterator it1;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    int x1;
    cout << "\nChoose which number you should remove: ";
    cin >> x1;
    ms1.erase(x1);

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    return 0;
}
```

```
================================================================================
Output:
Enter number of elements to store in multiset: 5
Enter 5 numbers: 1 5 2 1 1
1 1 1 2 5
Choose which number you should remove: 1
2 5
```

9. Create a c++ program using multiset and remove only one instance of element from multiset having same value.

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

int main()
{
    multiset<int> ms1;
    int number, x;
    cout << "Enter number of elements to store in multiset: ";
    cin >> number;
    cout << "Enter " << number << " numbers: ";
    for(int i = 0; i < number; i++)
    {
        cin >> x;
        ms1.insert(x);
    }

    multiset<int, greater<int>>:: iterator it1;

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    int x1;
    cout << "\nChoose which number you should remove: ";
```

```
    cin >> x1;
    ms1.erase(ms1.find(x1));

    for(it1 = ms1.begin(); it1 != ms1.end(); it1++)
        cout << *it1 << " ";

    return 0;
}
```

```
Output:
Enter number of elements to store in multiset: 5
Enter 5 numbers: 5 4 2 1 5
1 2 4 5 5
Choose which number you should remove: 5
1 2 4 5
```

10. Unlike a set, a multiset may contain multiple occurrences of the same number. The multiset equivalence problem states to check if two given multisets are equal or not. For example let A = {1, 2, 3} and B = {1, 1, 2, 3}. Here A is set but B is not (1 occurs twice in B), whereas A and B are both multisets. More formally, “Are the sets of pairs defined as \(A' = \{ (a, frequency(a)) | a \in \mathbf{A} \}\) equal for the two given multisets?” Given two multisets A and B, write a program to check if the two multisets are equal.

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

int main()
{
    multiset<int> ms1 = {1, 2, 3};
    multiset<int> ms2 = {1, 1, 2, 3};

    multiset<int>::iterator it1, it2;
    for (auto it1 : ms1)
        cout << it1 << " ";
    cout << endl;
    for (auto it2 : ms2)
        cout << it2 << " ";

    if (ms1 == ms2)
        cout << "\nBoth are equal";
    else
        cout << "\nBoth are not equal";

    return 0;
}
```

```
Output:
1 2 3
1 1 2 3
Both are not equal
```