STL - Sets and Maps

June 2017

Pre-requisites

- STL - Vectors and pairs
- Iterators

1 Sets and Maps and sort()

You can read the following article once you are thorough with the aforementioned concepts. Link to the artice : Sets and Maps and sort().

2 Some algorithms provided in STL

We will be discussing majorly two algorithms here which are closely related to sets or a sorted data structure per say. To use these algorithms, we need to include the following header file - algorithm. Thus the following line will solve our purpose

```
#include <algorithm>
```

1. lower_bound(starting iterator, ending iterator, key value)
   Let us say that our key value is k.
   The above algorithm will find the first occurrence of the element that is greater than or equal to k in the range of [starting iterator, ending iterator). The algorithm returns an iterator to the solution. Let us consider an exampe.
   NOTE: The range betweent the first iterator and second iterator must be sorted, otherwise this algorithm produces random results.

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int arr[] = {3, 4, 5, 5, 6, 6, 7, 8, 9};
vector<int> V(arr, arr + 9);      // note that the vector is sorted.

int main()
{
        cout << lower_bound(V.begin(), V.end(), 5) - V.begin();
//this will output 2 since 5(>= 5) occurs first at index 2.
}
```

Corner Case: When the key value is greater than the greatest value in the range, it returns end iterator.

2. upper_bound(starting iterator, ending iterator, key value)
   Again, let us say that our key value is k.
   The above algorithm will find the first occurrence of the element that is strictly greater than k in the range of [starting iterator, ending iterator). The algorithm returns an iterator to the solution. Let us consider an example:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

int arr[] = {3, 4, 5, 5, 6, 6, 7, 8, 9};
vector<int> V(arr, arr + 9);        // note that the vector is sorted.

int main()
{
        cout << upper_bound(V.begin(), V.end(), 5) - V.begin();
//this will output 4 since 6(> 5) occurs first at index 4.
}
```