**Map ..** Artful

Functions :  
begin() – Returns an iterator to the first element in the map  
end() – Returns an iterator to the theoretical element that follows last element in the map  
size() – Returns the number of elements in the map  
max\_size() – Returns the maximum number of elements that the map can hold  
empty() – Returns whether the map is empty  
pair insert(keyvalue,mapvalue) – Adds a new element to the map  
erase(iterator position) – Removes the element at the position pointed by the iterator  
erase(const g) - Removes the key value ‘g’ from the map  
clear() – Removes all the elements from the map  
key\_comp() / value\_comp() – Returns the object that determines how the elements in the map are ordered (‘<' by default)  
find(const g) – Returns an iterator to the element with key value ‘g’ in the map if found, else returns the iterator to end  
count(const g) – Returns the number of matches to element with key value ‘g’ in the map  
lower\_bound(const g) – Returns an iterator to the first element that is equivalent to mapped value with key value ‘g’ or definitely will not go before the element with key value ‘g’ in the map  
upper\_bound(const g) – Returns an iterator to the first element that is equivalent to mapped value with key value ‘g’ or definitely will go after the element with key value ‘g’ in the map.

void printMap(map <int, int> gquiz1)

{

/\*map <int, int> :: iterator itr;

for (itr = gquiz1.begin(); itr != gquiz1.end(); ++itr)

{

cout<<'\t'<< itr->first<<'\t'<< itr->second <<endl;

}\*/

/// OR ,

for (auto& x: gquiz1)

{

cout << x.first << ": " << x.second << '\n';

}

cout << endl;

}

int main()

{

map<string,int> mymap = {{ "alpha", 0 },{ "beta", 0 },{ "gamma", 0 } };

mymap.at("alpha") = 10;

mymap.at("beta") = 20;

mymap["gamma"] = 30;

for (auto& x: mymap)

{

cout << x.first << ": " << x.second << '\n';

}

map <int, int> gquiz1; // empty map container

gquiz1.insert(pair <int, int> (1, 40));

gquiz1.insert(pair <int, int> (2, 30));

gquiz1.insert(pair <int, int> (3, 60));

gquiz1.insert(pair <int, int> (4, 20));

gquiz1.insert(pair <int, int> (5, 50));

gquiz1.insert(pair <int, int> (6, 50));

gquiz1.insert(pair <int, int> (7, 10));

cout << "\nThe map gquiz1 is : \n";

cout << "\tKEY\tELEMENT\n";

printMap(gquiz1);

// assigning the elements from gquiz1 to gquiz2

map <int, int> gquiz2(gquiz1.begin(), gquiz1.end());

cout << "The map gquiz2 after assign from gquiz1 is : \n";

cout << "\tKEY\tELEMENT\n";

printMap(gquiz2);

// remove all elements up to element with key=3 in gquiz2

cout << "gquiz2 after removal of elements less than key=3 : \n";

cout << "\tKEY\tELEMENT\n";

gquiz2.erase(gquiz2.begin(), gquiz2.find(3));

printMap(gquiz2);

int num; // remove all elements with key = 4

num = gquiz2.erase (4);

cout << "\ngquiz2.erase(4) : ";

cout << num << " removed \n" ;

cout << "\tKEY\tELEMENT\n";

printMap(gquiz2);

//lower bound and upper bound for map gquiz1 key = 5

cout << "gquiz1.lower\_bound(5) : " << " KEY = ";

cout << gquiz1.lower\_bound(5)->first <<" ";

cout << " ELEMENT = " << gquiz1.lower\_bound(5)->second << endl;

cout << "gquiz1.upper\_bound(5) : " << " KEY = ";

cout << gquiz1.upper\_bound(5)->first <<" ";

cout << " ELEMENT = " << gquiz1.upper\_bound(5)->second << endl;

return 0;}

Output

alpha: 10

beta: 20

gamma: 30

The map gquiz1 is :

KEY ELEMENT

1: 40

2: 30

3: 60

4: 20

5: 50

6: 50

7: 10

The map gquiz2 after assign from gquiz1 is :

KEY ELEMENT

1: 40

2: 30

3: 60

4: 20

5: 50

6: 50

7: 10

gquiz2 after removal of elements less than key=3 :

KEY ELEMENT

3: 60

4: 20

5: 50

6: 50

7: 10

gquiz2.erase(4) : 1 removed

KEY ELEMENT

3: 60

5: 50

6: 50

7: 10

gquiz1.lower\_bound(5) : KEY = 5 ELEMENT = 50

gquiz1.upper\_bound(5) : KEY = 6 ELEMENT = 50