#include <iostream>

#include <conio.h>

#include <vector>

#include <algorithm>

#include <set>

#include <map>

#include <unordered\_map>

using namespace std;

bool f(int x, int y)

{

return x > y;

}

void vectordemo()

{

// in vector insert op and sorting takes big O(nlogn) time then we performs queries

vector<int> v1{ 11,2,3,44 };

cout << v1[0] << endl;

sort(v1.begin(), v1.end()); // (n logn) that is mergesort

for (int i = 0; i < v1.size(); i++)

{

cout << v1[i] << " ";

}

bool present = binary\_search(v1.begin(), v1.end(), 3); // logn times

bool present1 = binary\_search(v1.begin(), v1.end(), 33); // logn times

cout << present << endl; // true

cout << present1 << endl; // false

v1.push\_back(100);

v1.push\_back(100);

v1.push\_back(100);

v1.push\_back(113);

//for (int i = 0; i < v1.size(); i++)

//{

// cout << v1[i] << " ";

//}

// using iterator we print vector

// vector<int>::iterator p1 = lower\_bound(v1.begin(), v1.end(), 100); // >= big O(logn) times

// vector<int>::iterator p2 = upper\_bound(v1.begin(), v1.end(), 100); // > big O(logn) times

/\*while (p1 != v1.end())

{

cout << \*p1 << " ";

p1++;

}

while (p2 != v1.end())

{

cout << \*p2 << " ";

p2++;

}

\*/

//cout << endl;

//cout << \*p1 << " " << \*p2 << endl;

//cout << "diff " << p2 - p1;

//cout << endl;

sort(v1.begin(), v1.end(), f); // sort in desending order f is function

// using iterator we print vector

/\*vector<int>::iterator it3;

for (it3 = v1.begin(); it3 != v1.end(); it3++)

{

cout << \*it3 << " ";

}\*/

// using auto keyword we print vector

/\*for (int &x : v1)

{

x++;

}

for (int x : v1)

{

cout << x << " ";

}\*/

//auto p1 = lower\_bound(v1.begin(), v1.end(), 100); // >=

//auto p2 = upper\_bound(v1.begin(), v1.end(), 100); // >

//cout << endl;

//cout << \*p1 << " " << \*p2 << endl;

//cout << "diff " << p2 - p1;

//cout << endl;

}

void SetDemo()

{

set<int> S; // it takes in assending order

S.insert(-1); // insert operation takes big O(logn)

S.insert(12);

S.insert(-11);

S.insert(0);

S.insert(4);

S.insert(8);

cout << S.size() << endl;

auto it3 = S.begin();

for (it3 = S.begin(); it3 != S.end(); it3++)

{

cout << \*it3 << " ";

}

//auto it = S.find(-1); // if its not found it return last element

//if (it == S.end())

//{

// cout << "not present";

//}

//else

//{

// cout << "present : " << \*it;

//}

//auto it1 = S.upper\_bound(1);

//auto it2 = S.upper\_bound(2);

//cout << " Upperbound " << \*it1 << " " << \*it2;

//auto it3 = S.upper\_bound(12);

//if (it3 == S.end())

//{

// cout << "\nsorry not found: ...";

//}

}

void SetMap()

{

map<int,int> map1; // for add Big O (logn) times

map1[1] = 100;

map1[2] = 200;

map1[3] = 300;

map1[4] = 400;

for (auto it = map1.begin(); it != map1.end(); ++it)

{

cout << it->first << ":" << it->second << "\n";

}

map<char, int> map2; // n = size of string and for add/delete op logn = nlogn

string name = "jahanzaibshahid";

for (char c : name)

{

map2[c] = map2[c] + 1;

}

cout << map2['a'] << " " << map2['i'];

}

void UnorderedMap()

{

map<char, int> map2; // n = size of string and for add/delete op logn = O(nlogn)

string name = "jahanzaibshahid";

for (char c : name)

{

map2[c] = map2[c] + 1;

}

unordered\_map<char, int> Umap; // n = size of string and for add/delete op (1) = (n)

string name = "jahanzaibshahid";

for (char c : name)

{

Umap[c] = Umap[c] + 1;

}

}

void powerstl()

{

set<pair<int, int>> interval;

interval.insert({ 2,3 }); // pair {x,y} = {1,2}

interval.insert({ 10,20 });

interval.insert({ 30,400 });

interval.insert({ 401,450 });

int point = 1;

auto it = interval.upper\_bound({ point , INT\_MAX });

if (it == interval.begin())

{

cout << "its not lying in interval";

return;

}

it--;

pair<int, int> current = \*it;

if (current.first <= point && current.second >= point)

{

cout << "yes it exist " << current.first << " "<< current.second << endl;

}

else

{

cout << "its not lying in interval";

}

}

int main()

{

powerstl();

cout << endl;

return 0;

}