#include <bits/stdc++.h>

#include <iostream>

#include <vector>

#include <utility> //pair

#include <queue>   //Queue,priority\_queue

#include <deque>   //deque

#include <stack>

#include <map>

void fast()

{

    ios\_base::sync\_with\_stdio(0);

    cin.tie(nullptr);

    cout.tie(nullptr);

}

using namespace std;

int main()

{

    fast();

// vector

    vector<int> num0;

    vector<int> num1 = {10, 20, 30, 555};

    vector<int> num3(4, 50);

    vector<int> num4(4); // size element

    num0.push\_back(50);

    num0.push\_back(40);

    //    num0.at(1)=5;//error

    for (int i = 0; i < num0.size(); i++)

        cout << num0.at(i) << " "; // 50,40

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n";

    for (int i = 0; i < num1.size(); i++)

        cout << num1.at(i) << " "; // 10 20 30 555

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n";

    for (int i = 0; i < num3.size(); i++)

        cout << num3.at(i) << " "; // 50 50 50 50

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n";

    cout << "number of num3: " << num3.size(); // 4

    num3.push\_back(100);

    cout << "\nnumber of num3: " << num3.size(); // 5

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n";

    for (int i = 0; i < num3.size(); i++)

        cout << num3.at(i) << " "; // 50 50 50 50 100

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n";

    cout << "number of num1: " << num1.size(); // 4

    num1.pop\_back();

    cout << "\nnumber of num1: " << num1.size() << "\n"; // 3

    for (int i = 0; i < num1.size(); i++)

        cout << num1.at(i) << " "; // 10 20 30

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n";

    cout << num1.front() << endl;                        // 10

    cout << num1.back();                                 // 30

    cout << "\nnumber of num1: " << num1.size() << "\n"; // 3

    for (int i = 0; i < num1.size(); i++)

        cout << num1.at(i) << " ";                 // 10 20 30

    vector<vector<int>> hi(5, vector<int>(10, 0)); // 5 row 10 coulm

// pair

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n\n";

    pair<char, int> pair2('a', 10);

    pair<char, int> pair3 = make\_pair('a', 10);

    pair<int, int> pair4;

    pair<int, vector<int>> pair5;

    pair<int, pair<int, string>> pair6;

    vector<pair<int, string>> v;

    v.emplace\_back(2, "");

    v.push\_back(make\_pair(0, ""));

    pair4.first = 2;

    pair4.second = 6;

    pair5.first = 7;

    pair5.second.push\_back(66);

    pair5.second.push\_back(65);

    pair4.first = 2;

    pair6.second.first = 5;

    pair6.second.second = "gugubl";

    v.at(0).first = 10;

    v.at(0).second = "knoll";

    v.at(1).first = 10;

    v.at(1).second = "knoll";

    cout << pair5.second.at(0) << endl;  // 66

    cout << pair6.second.first << endl;  // 5

    cout << pair6.second.second << endl; // gugubl

    cout << v.at(0).first << endl;       // 10

    cout << v.at(0).second << endl;      // knoll

    cout << (pair2 == pair3) << endl;    // 1

// queue

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n\n";

    queue<int> q1; // empty queue of ints

    queue<string> q2({"apple", "banana", "orange"});

    // queue<int> q2 = {1,2,3,4,5}; // wrong way

    q1.push(5);

    q1.push(6);

    q1.push(9);

    cout << q1.front() << endl; // 5

    cout << q1.back() << endl;  // 9

    cout << q1.size() << endl;  // 3

    q1.pop();                   // delet 9

    cout << q1.front() << endl; // 6

    cout << q1.empty() << endl; // 0

    q1.emplace(20);

    cout << q1.back() << endl; // 20

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n\n";

// deque

    deque<int> dq; // Empty deque of ints

    deque<int> dq2 = {1, 2, 3, 4, 5};

    dq.push\_front(10);

    dq.push\_back(20);

    dq.push\_front(30);

    dq.push\_back(40);

    for (int x : dq)

    {

        cout << x << " "; // 30 10 20 40

    }

    cout << endl;

    dq.pop\_back();

    dq.pop\_front();

    for (int x : dq)

    {

        cout << x << " "; // 10 20

    }

    cout << endl;

    dq.assign(5, 6);

    for (int x : dq)

    {

        cout << x << " "; // 6 6 6 6 6

    }

    cout << endl;

    dq.emplace(dq.cbegin() + 2, 5);

    for (int x : dq)

    {

        cout << x << " "; // 6 6 5 6 6 6

    }

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n\n";

// stack

    stack<int> st;

    st.push(50);

    st.push(60);

    st.push(90);

    int x = st.top();

    cout << st.size() << endl;

    cout << x << endl; // 90

    st.pop();

    x = st.top();

    cout << x << endl;          // 60

    cout << st.empty() << endl; // 0

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n\n";

// priority\_queue

    priority\_queue<int> pq;

    pq.push(5);

    pq.push(10);

    pq.push(9);

    pq.push(7);

    cout << "size:" << pq.size() << "\nelement pq:"; // 4

    while (!pq.empty())

    {

        cout << pq.top() << " "; // 10 9 7 5

        pq.pop();

    }

    cout << "\n======\n";

    priority\_queue<int, vector<int>, greater<int>> pq1; // عكس pq

    pq1.push(5);

    pq1.push(10);

    pq1.push(9);

    pq1.push(7);

    cout << "element pq1:";

    while (!pq1.empty())

    {

        cout << pq1.top() << " "; // 5 7 9 10

        pq1.pop();

    }

    cout << "\n======\n";

    priority\_queue<int> pq2;

    pq2.push(55);

    pq2.push(55);

    pq2.push(55);

    pq2.push(55);

    pq2.push(55);

    pq.swap(pq2);

    while (!pq.empty())

    {

        cout << pq.top() << " "; // 55 55 55 55 55

        pq.pop();

    }

    cout << "\n======\n";

    while (!pq2.empty())

    {

        cout << pq2.top() << " "; // 10 9 7 5

        pq2.pop();

    }

    cout << "\n\*\*\*\*\*\*\*\*\*\*\*\n\n";

// map

    map<int, int> m; //<key,value>

    m[1] = 5;        // key=>1   value=>5

    m[10] = 54;

    m[5];

    m[0] = 6;

    int x = 8, y = 77;

    m[x] = y;

    m.insert(pair<int, int>(4, 50)); // key=>4  value=>50

    m.insert(pair<int, int>(1, 50)); // not insert key really exist

    for (auto it : m)

   cout<<"key"<<it.first<<"value"<<it.second<<endl;// print sort key (0,1,4,5,10)

    if (m.find(10) != m.end()) // 10=>key   exist

        cout << "true";

    if (m.find(2) == m.end()) // not exist

        cout << "\nfalse";

    if (m.count(10)) // 10=>key   exist

        cout << "\ntrue";

    if (!m.count(2)) // not exist

        cout << "\nfalse";

    // m.count() return 0||1

    cout << "\nsize: " << m.size();

    m.erase(x);

    cout << "\nsize: " << m.size() << endl;

    m.erase(m.begin() /\*|| m.cbegin() \*/); // delete first element

    for (auto it : m)

        cout<<"key"<<it.first <<"value"<<it.second<<endl; // print key (1,4,5,10)

    m.erase(prev(m.end() /\*|| m.cend()\*/));                             // delete last element

    cout << "\*\*\*\*\*\*\*\*\*\n";

    for (auto it : m)

        cout<<"key"<<it.first<<"value"<<it.second<<endl;// print sort key (1,4,5)

    cout << "empty: " << m.empty();                                     // 0

    m.clear();                                           // delete all element

    cout << "\nempty: " << m.empty();                                   // 1

    unordered\_map<int, int> m1;

    m1[5] = 4;

    m1[4] = 40;

    m1[10] = 24;

    for (auto it : m1)

        cout<<"key:"<<it.first <<"value:"<<it.second<<endl; // print key (10,4,5)

//set

    set<int> s1; // sort smallest to greater

    s1.insert(5);

    s1.insert(5);

    s1.insert(4);

    s1.insert(50);

    s1.insert(9);

    s1.insert(15);

    for (auto it : s1)

        cout << "value: " << it << endl; // print  (4,5/\*one only\*/,9,15,50)

    cout << \*s1.lower\_bound(15) << endl; // after x or equal

    cout << \*s1.upper\_bound(15) << endl; // after x

    cout << "\*\*\*\*\*\n";

    set<int, greater<>> s2; // sort greater to smallest

    s2.insert(5);

    s2.insert(4);

    s2.insert(50);

    s2.insert(9);

    s2.insert(15);

    s2.lower\_bound(9);

    for (auto it : s2)

        cout << "value: " << it << endl; // print  (50,15,9,5,4)

    if (s2.find(5) != s2.end())          // exist

        cout << "true\n";

    if (s2.find(44) == s2.end()) // not exist

        cout << "false\n";

    if (s2.count(5)) // exist

        cout << "true";

    if (!s2.count(44)) // not exist

        cout << "\nfalse\n";

    s2.erase(s2.find(5));

    for (auto it : s2)

        cout << "value: " << it << endl; // print  (50,15,9,4)

    cout << \*s2.lower\_bound(50) << endl; // after x or equal

    cout << \*s2.upper\_bound(50) << endl; // after x

    multiset<int> ms;

    ms.insert(5);

    ms.insert(5);

    ms.insert(4);

    ms.insert(4);

    ms.insert(9);

    ms.insert(15);

    for (auto it : ms)

        cout << "value: " << it << endl; // print  (4,4,5,5,9,15)

    cout << "\*\*\*\*\*\n";

    multiset<int, greater<>> ms1;

    ms1.insert(5);

    ms1.insert(5);

    ms1.insert(4);

    ms1.insert(4);

    ms1.insert(9);

    ms1.insert(15);

    for (auto it : ms1)

        cout << "value: " << it << endl; // print  (15,9,5,5,4,4)

    cout << \*ms1.begin() << endl;        // first element

    cout << \*ms1.crbegin() << endl;      // last element

}

/\*

    push\_back(5);بتخزن القيمه بعد اخر قيمه

    =

    emplace\_back(2);faster

    insert(iterator,value)بيخزن قيمه في مكان البشاور عليه

    insert(iterator,value1,value2) يخزن قيمه2 في مكان البشاور عليه بعدد قيمه1

    iterator=begin()||end()||r

    pop\_back()بتشيل اخر عنصر

    erase(iterator)بتشيل الرقم المشار عليه

    erase(iterator1,iterator2 +1)بيشيل الارقام من iterator1الي iterator2

    v1.swap(v2)v1بيتبدل مع v2

    auto iter = num.rbegin();||vector<int>::reverse\_iterator iter=num.rbegin();

    auto iter = num.rend();||vector<int>::reverse\_iterator iter=num.rend();

    rbegin=end-1

    rbegin++=end-2

    rend-1=begin

    rend--=begin+1

    vector<int>::iterator last = num1.end() - 1; // بشاور علي اخر عنصر

int gcd(int n1, int n2) {

    if (n2 != 0)

        return gcd(n2, n1 % n2);

    else

        return n1;

}

int rows[5] = {11, 22, 33, 44, 55};

    int \*ptr = rows;

    int \*pt = new int(20);

    cout<<pt<<"\n";//address 20

    cout<<\*pt<<"\n";//value 20

    cout<<&pt<<"\n";//address pt

    for (int i = 0; i < 5; i++)

    {

        cout << \*rows + i << "\n";

    }

    for (int i = 0; i < 5; i++)

    {

        cout << \*(rows + i) << "\n";

    }

    for (int i = 0; i < 5; i++)

    {

        cout << \*(ptr + i) << "\n";

    }

    for (int i = 0; i < 5; i++)

    {

        cout << \*ptr + i << "\n";

    }

\*/