```
/*
    Lý thuyết về Thư viện STL (Standard Template Library)
    - Là thư viện chứa những template của các cấu trúc dữ liệu cũng như thuật toán tổng quát hỗ trợ người dùng
    - Sức mạng của C++ là Thư viện STL.
    - Thư viện STL giúp người dùng thực hiện các công việc
    như ra vào dữ liệu, quản lý mảng động, xây dựng sẵn những cấu trúc dữ liệu cơ bản (NGĂN XẾP, HÀNG ĐỢI, TẬP HỢP, ...)
    các thuật toán sắp xếp, tìm min - max hoặc ước chung nhỏ nhất

    - CÁC THÀNH PHẦN CHÍNH CỦA THƯ VIỆN STL TRONG C++
        1. CONTAINERS Library:
        - Chứa các kiểu cấu trúc dữ liệu mẫu như
            + vector
            + set
            + map
            + stack
            + queue
            ...
        2. ALGORITHM Library:
        - Chứa các thuật toán viết sẵn để thao tác với dữ liệu như
            + sort
            + reverse
            + *max_element
            + *min_element
            + binary_research
            ...
        3. ITERATOR Library:
        - Là các biến lặp, sử dụng truy cập, duyệt các phần tử dữ liệu của các containers.
        - Về cơ bản, nó giống như các biến chạy trên dữ liệu nhưng truy cập vào địa chỉ của dữ liệu
        - Dễ hình dung nhất là con trỏ (Pointer)
            ...
```

Vector_Iterator.cpp 1

STL_01 > Vector_Iterator.cpp > ...

Click here to ask Blackbox to help you code faster

```cpp
#include <iostream>
#include <vector>
using namespace std;
    /*
    VECTOR
        - Cú pháp:
        vector <data type> name
        - in ra các phần tử trong vecor giống như in ra các phần tử trong mảng duyệt qua
        - chỉ số của từng element trong vector bắt đầu từ chỉ số (index) = 0
        - Cách duyệt vector: có 3 cách
        * C1: giống array
            for (int i = 0; i < v.size(); i++)
                cout << v[i] << " ";
        * C2: for each
            for (int x : v)
                cout << x << " ";
        * C3: Duyệt bằng iterator (giống con trỏ)
            for (vector<int> :: iterator name_it = v.begin(); name_it != v.end(); name_it
                cout << *name_it << " ";
```

```
MỘT SỐ HÀM TRONG VECTOR
1/ push_back
    - Đẩy phần tử vao cuối dánh sách vector
    - Cú pháp:
        v.push_back(value)
2/ pop_back
    - Xóa phần tử ở cuối
    - Cú pháp:
        v.pop_back()
3/ size
    - Trả về số lượng element trong vecor
    - Cú pháp:
        v.size()
4/ erase
    - Xóa phần tử tại vị trí muốn xóa trong khoảng ví trí bắt đầu tới vị trí cuối
    - Cú pháp:
    v.erase(vị trí bắt đầu, vị trí cuối)
    vd:
        0 1 2 3 4 5 6
        v.erase(v.begin(), v.begin() + 2)
        0 2 3 4 5 6
5/ insert
    - chèn phần tử trong vector
    - Cú pháp:
        v.insert(iterator, value)
6/ v.begin()
    - vị tr element đầu tiên của vector
    - Cú pháp:
        cout << *v.begin();
        vì v.begin() giống iterator là con trỏ
7/ v.end()
    - vị trí sau vị trí element cuối của vector
    - Cú pháp:
        cout << *(v.end() - 1);
8/ v.empty()
    - Không có element => return 1
    - Có element => return 0
*/
```

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main()
{
    vector <int> v;
    v.push_back(3);
    v.push_back(6);
    v.push_back(5);
    v.push_back(1);
    v.push_back(12);
    v.push_back(10);
    // các element ban đầu trong vector
    // 3 6 5 1 12 10
    v.insert(v.begin() + 1, 4);
    // 3 4 6 5 1 12 10
    v.erase(v.begin() + 1, v.begin() + 2);
    // 6 5 1 12 10
    // duyệt qua các phần tử trong vector bằng for each
    for (int x : v)
        cout << x << " ";
    cout << endl;
    // sử dụng reverse để lật ngược các element trong vector
    // hàm reverse trong STL không trong vector
    reverse(v.begin(), v.end());
    for (int x : v)
        cout << x << " ";
    return 0;
}
```

Tron2MangSapXep_Vector.cpp

STL_02 > Tron2MangSapXep_Vector.cpp > ...

Click here to ask Blackbox to help you code faster

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main()
{
    int n, m;
    cin >> n >> m;
    int a[1000], b[1000];
    for (int i = 0; i < n; i++)
        cin >> a[i];
    for (int j = 0; j < m; j++)
        cin >> b[j];
    int i = 0, j = 0;
    // nếu không xài vector thì phải có h = 0, hop[500 + 500]
    // Tạo 1 vector
    vector<int> hop;
    while (i < n && j < m)
    {
        if (a[i] <= b[j])
        {
            hop.push_back(a[i]);
            i++;
        }
        else
        {
            hop.push_back(b[j]);
            j++;
        }
    }
    // kiểm tra xem 1 trong 2 mảng còn phần tử nào thì xếp vào mảng đang sắp xếp
    while (i < n)
    {
        hop.push_back(a[i]);
        i++;
    }
    while (j < m)
    {
        hop.push_back(b[j]);
        j++;
    }
    // auto sẽ tự động chọn kiểu dữ liệu phụ hợp với array
    for (int x : hop)
        cout << x << " ";
    return 0;
}
```

Pair.cpp

STL_03 > Pair.cpp > ...

Click here to ask Blackbox to help you code faster

```
/*
    Pair:
    - Khái báo thư viện
    #include <utility>
    - Lưu 1 cặp gái trị có cặp giá trị giống hoặc khác nhau
    - dùng để kết hợp với vector, set, map
    - first là tham chiếu trực tiếp vào phần tử đầu tiêm
    - second là tham chiếu trực tiếp vào phần tử thứ 2
    - Cú pháp:
        pair<data_type1, data_type2> p{value1, value2}
    - in các giá trị của pair
    cout << p.first hoặc p.second
    - CÁC HÀM TRONG PAIR
        1/ Swap(pair): hoán đổi 2 pair với nhau
        - các data type của first và second của 2 pair hoán đổi nhau phải cùng data type thì mới swap được
*/
```

```
18  /*      VD0:
19  Pair kết hợp vector
20  int main()
21  {
22      vector<pair<int, int>> v;
23      v.push_back({ 1, 2 });
24      v.push_back({ 3, 4 });
25      v.push_back({ 4, 6 });
26      v.push_back({ 9, 12 });
27
28      for (int i = 0; i < v.size(); i++)
29          cout << v[i].first << " " << v[i].second << endl;
30      return 0;
31  }
32  */
```

```
/*      VD1:
pair <int, int> p;
hoặc pair <int, int> p(value1 ,value2);
// có thể gán giá trị vào p.first và p.second
p.first = 3;
p.second = 6;
// in ra pair giống vector
cout << p.first << " " << p.second;
*/
```

```
43
44    /*      VD2:
45    pair <string, int> p1("Nguyen Van A", 20);
46    pair <string, int> p2("Nguyen Van B", 19);
47    // sử dụng hàm swap để thay đổi 2 cái pair cho nhau
48    p1.swap(p2);
49    cout << p1.first << " " << p1.second << endl;
50    cout << p2.first << " " << p2.second << endl;
51    */
52
```

```
/*      VD3
    int n;
    cout << "Nhap so cap: ";
    cin >> n;
    vector<pair<int, int>> v;
    // Nhập cặp giá trị
    for (int i = 0; i <= n; i++)
    {
        int x, y;
        cout << "Nhap x: ";
        cin >> x;
        cout << "Nhap y: ";
        cin >> y;
        v.push_back({ x, y });
    }
    // in ra cặp
    // theo chỉ số
    cout << "In ra bang for binh thuong: " << endl;
    for (int i = 0; i < v.size(); i++)
    {
        cout << "Cap v[" << i << "]: ";
        cout << v[i].first << " " << v[i].second << endl;
    }
    cout << "In ra bang for each:" << endl;
    // Cách 2 xuất ra bảng for each
    for (pair<int, int> x : v)
        cout << x.first << " " << x.second << endl;
    */

```

```cpp
#include <iostream>
#include <vector>
#include <utility>
using namespace std;

int main()
{
    // muốn xuất ra 3 value trong pair
    int n;
    cout << "Nhap so cap: ";
    cin >> n;
    // tạo vecor chứa cặp 3 số
    vector < pair<pair <int, int>, int>> v;
    v.push_back({ {1, 2}, 3 });
        // in ra bằng auto
    cout << "In ra bang for auto" << endl;
    for (auto x : v)
    {
        // x.first: pair <int, int>
        cout << x.first.first << " " << x.first.second << " " << x.second;
    }
    return 0;
}
```

ChenPTuVaoMang_Vetor.cpp

STL_04 > ChenPTuVaoMang_Vetor.cpp > ...

Click here to ask Blackbox to help you code faster

```cpp
#include <iostream>
#include <vector>
#define ll long long
using namespace std;
    // Buổi 7_Bài 14: Chèn 1 phần tử vào trong mảng
int main()
{
    int n, x, index;
    cin >> n >> x >> index;
    // chèn phần tử x vào index mà index start from 1
    // tạo 1 vector
    vector <int> v;
    // Nhập các phần tử vào vector => Chỉ dùng biến temp và chỉ sử dụng hàm push_back
    for (int i = 0; i < n; i++)
    {
        int temp;
        cin >> temp;
        v.push_back(temp);
    }
    // chèn phần tử x
    // v.insert(vị trí chèn, value)
    v.insert(v.begin() + index - 1, x);
    for (int i = 0; i < v.size(); i++)
        cout << v[i] << " ";
    return 0;
}
```

Map.cpp

STL_05 > Map.cpp > main()

Click here to ask Blackbox to help you code faster

```
/*
    Lý thuyết về MAP
    - Map là container giúp lưu các phần tử  theo cặp key, value (khóa-giá trị)
    - Mỗi giá trị của key sẽ ánh xạ sang 1 value tương ứng
    - So với Set thì Map còn mạnh hơn và giải quyết được nhiều vấn đề
    - Ứng dụng
        + Các key trong map là những giá trị riêng biệt
        + Không có 2 ket nào có giá trị giống nhau, value thì có thể trùng nhau
        + Map tìm kiếm nhanh
        + Map có thể dùng key làm index để truy cập vào value tương ứng
    - Khai báo sử dụng thứ viện map
        #include <map>
    - Map có 3 loại
    1/ MAP
        - Khái báo:
            map <key_data_type, value_data_type> map_name;
        - Các cặp phần tử trong map được sắp xếp theo thứ tự TĂNG DẦN của KEY
        - Mỗi phần tử trong map thực chất là 1 pair với:
            + first lưu key
            + second luu value

```

```
- MỘT SỐ HÀM THÔNG DỤNG TRONG MAP
1/ insert
    - Thêm 1 phần tử trong map
    - Cú pháp:
        map[key] = value hoặc
        map.insert({key, value})
2/ size
    - Trả về số lượng phần tử trong map
    - Cú pháp:
        map.size()
3/ erase
    - Xóa 1 phần tử trong map với độ phức tạp O(logN)
    - Trước khi dùng hàm erase phải đảm bảo phần tử cần xóa tồn tại trong map nếu không => xảy ra lỗi RUNTIME ERROR
    - xóa key
    - Cú pháp:
        map.erase(value)
4/ clear
    - Xóa mọi phần tử trong map
    - Cú pháp:
        map.clear()
5/ empty
    - Kiểm tra map Rỗng
        + Nếu rỗng => return true
        + Nếu không rỗng => return false
    - Cú pháp:
        map.empty()
6/ find
    - Tìm kiếm sự xuất hiện của 1 key nào đó trong map
    - Độ phức tạp O(logN)
    - Hàm này trả về iterator tới cặp phần tủ nếu tìm thấy, ngược lại nó trả về iterator end() của map khi khi giá trị key tìm kiếm không tồn tại trong map
    - Cú pháp:
        map.find(key)
    - Cách duyệt
        if (mp.find(key) != mp.end())
            cout << (*mp.find(key)). second;
            // second là value tương ứng với key mình cần tìm
```

```
7/ cout
    - Hàm này để đếm số lần xuất hiện của 1 key trong map
    - giá trị trả về 0 hoặc 1
    - có thể thay thế hảm find
    - Độ phức tạp O(logN)
    - Vd
        if (mp.count(key) != 0)
            cout << "YES";
        else
            cout << "NO";
ULTIMAP
- Khai báo: multimap<ket_data_type, value_data_type> multimap_name
- Tương tự như map
- Khác biệt TỒN TẠI NHIỀU KEY CÓ CÙNG GIÁ TRỊ
- Tính chát key sắp xếp tăng dần, hay độ phức tạp, cách dùng đều giống như map
- Có 3 HAM2khác biệt: find, cont, erase

- MỘT SỐ HÀM TRONG MULTIMAP
1/ find
    - vì trong multimap có nhiều key có cùng giá trị nếu ta tìm kếm 1 key nào đó sẽ trả về vị trí xuất hiện đầu tiên của key đó trong mao
    - Cú pháp:
        map.find(key)
    - Ví dụ
    cout << (*mulmp.find(key)).second;
    => second là trả về value đầu tiên
3/ Hảm erase
    - cout << (mulmp.count(key));
2/ erase
    - Xóa thông qua giá sẽ xóa tất cả các key tương ứng
    - Cú pháp:
        map.erase(value)
```

```
3/ UNORDEREDMAP

- Cách duyệt Map: 2 cách duyệt
    cách 1: for-each
        for (pair<int, int> it : mp)
            cout << it.first << " " << it.second << endl;
    Cách 2: for each dùng auto thay cho pair
        for (auto it : mp)
            cout << it. first << " " << it.second << endl;
```

```cpp
    // MAP
/*      VD1: Hàm insert
    // Khai báo map
    map<int, int> mp;
    mp.insert({ 1, 2 });
    mp.insert({ 1, 3 });
    mp.insert({ 1, 4 });
    mp.insert({ 1, 10 });
    mp.insert({ 1, 9 });
    // số lượng phần tử trong map => 1 phần tử
    // cout << mp.size(); // kết quả: 1
    // Cách 2 đẩy trả trị vào map
    // mp[key] = value
    mp[3] = 2;
    // key có thể âm
    cout << mp.size(); // kết quả: 2 vì có 2 key
    // nếu key giống nhau thì chỉ lấy giá trị cuối cùng đẩy vào
*/

/*      VD2: Hàm erase
    map<int, int> mp;
    mp.insert({ 1, 2 });
    mp.insert({ 7, 3 });
    mp.insert({ 3, 4 });
    mp.insert({ 2, 10 });
    mp[1] = 1;
    // Xóa các key
    // mp.erase(key)
    mp.erase(1);
    for (auto x : mp)
        cout << x.first << " " << x.second << endl;
*/
```

```cpp
/*      VD3: Hàm clear: xóa hết
    map<int, int> mp;
    mp.insert({ 1, 2 });
    mp.insert({ 7, 3 });
    mp.insert({ 3, 4 });
    mp.insert({ 2, 10 });
    mp[1] = 1;
    // Xóa các key
    mp.clear();
    for (auto x : mp)
        cout << x.first << " " << x.second << endl;
*/

/*      VD4: Hàm empty
    map<int, int> mp;
    mp.insert({ 1, 2 });
    mp.insert({ 7, 3 });
    mp.insert({ 3, 4 });
    mp.insert({ 2, 10 });
    mp[1] = 1;
    mp.clear();
    cout << mp.enpty();
    // nếu map rỗng => 1
    // Nếu map không rỗng => 0
*/

/*      VD5: Hàm find
    map<int, int> mp;
    mp.insert({ 1, 2 });
    mp.insert({ 7, 3 });
    mp.insert({ 3, 4 });
    mp.insert({ 2, 10 });
    mp[1] = 1;
    // tìm kiếm phần tử có tồn tại hay không và tìm theo key, độ phức tạp O(logN)
    // cách duyệt
    if (mp.find(3) != mp.end())
        cout << (*mp.find(3)).second;
*/
```

```cpp
/*      VD6: Hàm count
    map<int, int> mp;
    mp.insert({ 1, 2 });
    mp.insert({ 7, 3 });
    mp.insert({ 3, 4 });
    mp.insert({ 2, 10 });
    mp[1] = 1;
    // hàm count
    if (mp.count(7) != 0)
        cout << "YES";
    else
        cout << "NO";
*/

    // MULTIMAP
/*      VD1: cách khai báo, duyệt, chèn trong hàm multimap
    // khai báo multimap
    multimap<int, int> mulmp;
    mulmp.insert({ 1, 2 });
    mulmp.insert({ 7, 3 });
    mulmp.insert({ 3, 4 });
    mulmp.insert({ 2, 10 });
    mulmp.insert({ 1, 4 });
    cout << mulmp.size() << endl;
    for (auto x : mulmp)
        cout << x.first << " " << x.second << endl;
*/
```

```
/*
    VD2: Hàm find
    multimap<int, int> mulmp;
    mulmp.insert({ 1, 2 });
    mulmp.insert({ 1, 7 });
    mulmp.insert({ 7, 3 });
    mulmp.insert({ 3, 4 });
    mulmp.insert({ 2, 10 });
    mulmp.insert({ 1, 4 });
    cout << (*mulmp.find(1)).second;
    // kết quả là 2
*/

/*      VD3: Hàm count
    multimap<int, int> mulmp;
    mulmp.insert({ 1, 2 });
    mulmp.insert({ 1, 7 });
    mulmp.insert({ 7, 3 });
    mulmp.insert({ 3, 4 });
    mulmp.insert({ 2, 10 });
    mulmp.insert({ 1, 4 });
    cout << (mulmp.count(1)) << endl;
    // đáp án là 3
    // kt tồn tại hay không

    if (mulmp.count(1) != 0)
        cout << "YES";
    else
        cout << "NO";
*/
```

```
/*      VD4: Hàm erase
    // Hàm erase xóa 1 phần tử cụ thể => phải chỉ rõ iterator của phần tử đó và duyệt qua iterator
    multimap<int, int> mulmp;
    mulmp.insert({ 1, 2 });
    mulmp.insert({ 1, 7 });
    mulmp.insert({ 7, 3 });
    mulmp.insert({ 3, 4 });
    mulmp.insert({ 2, 10 });
    mulmp.insert({ 1, 4 });
    // duyệt bằng iterator
    map<int, int> :: iterator it = mulmp.begin(); // nếu thêm it++ => xóa 1, 7
    mulmp.erase(it);
    // duyệt
    for (auto x : mulmp)
        cout << x.first << " " << x.second << endl;
    */
```

DemTanSuat_Map.cpp

STL_06 > DemTanSuat_Map.cpp > ...

Click here to ask Blackbox to help you code faster

```cpp
#include <iostream>
#include <map>
#define ll long long
using namespace std;
    // Bươi_ Bài11 đếm tần suất bằng MAP
    // map[a[i]]++
int main()
{
    map<ll, ll> mp;
    int n;
    cin >> n;
    int a[100000];
    // Nhập mảng
    for (int i = 0; i < n; i++)
    {
        cin >> a[i];
        // Nhập phần tử nào xong thì đẩy phần tử đó vào trong map và tăng tần suất lên
        // ban đầu là 1, 1 rồi tới 1, 2 rồi 1, 3 rồi lấy thằng cuối cùng
        // chỗ ++ là tăng value trong map
        mp[a[i]]++;
    }
    // Map sắp xếp sẵn key tăng dần tương ứng với giá trị phần tử từ nhỏ đến lớn
    for (auto x : mp)
        cout << x.first << " " << x.second << endl;
    cout << endl;
    for (int i = 0; i < n; i++)
        // làm như vậy sẽ bị lặp phần tử
        // vậy phải kt
        if (mp[a[i]] != 0)
        {
            cout << a[i] << " " << mp[a[i]] << endl;
            // sau khi thế rồi đẩy value =  vào lại map => tránh trường hợp lặp
            mp[a[i]] = 0;
        }

    return 0;
}
```

Sort.cpp

STL_07 > Sort.cpp > ...

Click here to ask Blackbox to help you code faster

```
/*
    Lý thuyết về Sort trong STL
    - Hàm sort trong STL thuộc thư viện <algorithm>
    - Độ phức tạp là O(log2(N))
    - Mấy hàm sort thông thường như:
        + selection sort, bubble sort, insertion sort, interchange sort, couting sort ...
        đều có độ phức tạp là O(N^2)
    - Đi làm thường xài hàm sort của STL
        - sử dụng for each để duyệt mảng lẹ
    - có thể sử dụng cho vector , string, array, ...

    - Hàm sort của STL thường có 3 parameter (tham số):
        * sort(a, a + n, cmp)
            + a: là mảng cần sort
            + a + n: là địa chỉ của phần tử sau phần tử cuối cùng của mảng
            + cmp: là hàm so sánh 2 phần tử, viết tắt của từ COMPARE

    - Hàm sort trong STL nếu SẮP XẾP TĂNG DẦN chỉ cần 2 paremeter (tham số)
        * sort(a + 0, a + n)
            + a + 0: là địa chỉ của phần tử đầu tiên của mảng và là phần tử xét đầu tiên
            + a + n: là địa chỉ của phần tử sau phần tử cuối cùng của mảng
            + a + n: là cận thì hàm sort sẽ không xét đến phần tử này

    - Hàm sort trong STL nếu SẮP XẾP GIẢM DẦN cần tới 3 paremeter (tham số)
        * sort(a + 0, a + n, MyFunction)
            + a + 0: là địa chỉ của phần tử đầu tiên của mảng và là phần tử xét đầu tiên
            + a + n: là địa chỉ của phần tử sau phần tử cuối cùng của mảng
            + MyFunction: thường là hàm TỰ VIẾT để so sánh các tiêu chí mình suy nghĩ ra để thực hiện với các phần tử
            + greater<data type>(): là hàm SẮP XẾP GIẢM DẦN có sẵn trong STl
            + thường hàm greater<data type>() được sử dụng để sort mảng giảm dần
            + MyFuntion: là hàm tự viết luôn có data type là bool thường đặt tên là cmp nghĩa là compare: so sánh
            + hàm MyFunction có 2 parameter (tham số) là 2 phần tử cần so sánh là bool(data type a, data type b)
                + Trong đó:
                - a: là phần tử đứng trước
                - b: là phần tử đứng sau
                - Giá trị trả về true: GIỮA NGUYÊN VỊ TRÍ
                - Giá trị trả về false: HOÁN VỊ GIÁ TRỊ 2 PHẦN TỬ
*/
```

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

bool cmp(int, int);

int main()
{
    // có 1 mảng a[] gồm 6 element(phần tử) ban đầu: 2, 3, 10, 1, 7, 2, 4
    int a[7] = { 2, 3, 10, 1, 7, 2, 4 };
    // sử dụng hàm sort trogn stl để sắp xếp mảng a[] tăng dần
    cout << "Ham sap xep tang dan trong STL: ";
    sort(a + 0, a + 7);
    for (int x : a)
        cout << x << " ";
    // Kết quả sau khi thuật toán sắp xếp được thực hiện: 1 2 2 3 4 7 10
    cout << endl;
    cout << "\nHam sap xep giam dan trong STL bang ham co san: ";
    // sắp xếp giảm dần trong mảng a[7] bằng cách sử dụng hàm sort trong stl và hàm có sẵn
    sort(a + 0, a + 7, greater<int>());
    for (int x : a)
        cout << x << " ";
    cout << "\n\nHam sap xep giam dan tu viet: ";
    // sắp xếp giảm dần cần tới 3 tham số
    sort(a + 0, a + 7, cmp);
    for (int x : a)
        cout << x << " ";
    return 0;
}

bool cmp(int a, int b)
{
    // kiểm tra nếu sắp xếp giảm dần
    if (a > b)
        return true; // không hoán vị giá trị đúng
    return false;   // hoán vị giá trị
}
```

SuDungHamSortTrongSTL_TheoYCau.cpp

STL_08 > SuDungHamSortTrongSTL_TheoYCau.cpp > ...

Click here to ask Blackbox to help you code faster

```cpp
/*
    Sử dụng hàm Sort trong thư viện STL theo yêu cầu.
    Cho mảng số nguyên A[] có N phần tử, hãy sắp xếp các phần tử trong mảng theo thứ tự tăng dần sau đó giảm dần.

    Input Format
    Dòng đầu tiên là số nguyên dương N.
    Dòng thứ 2 là N phần tử trong mảng, các phần tử viết cách nhau một dấu cách. (1<=N<=10^5;-10^9<=A[i]<=10^9)

    Output Format
    Dòng dầu tiên in ra mảng theo thứ tự tăng dần.Dòng hai in ra mảng theo thứ tự giảm dần

    Dữ liệu vào:
    5
    14 -88 6 23 -14
    Dữ liệu ra:
    -88 -14 6 14 23
    23 14 6 -14 -88
*/
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

bool cmp(long, long);

int main()
{
    int n;
    cin >> n;
    long a[100000];
    for (int i = 0; i < n; i++)
        cin >> a[i];

    sort(a + 0, a + n);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;

    sort(a + 0, a + n, cmp);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    return 0;
}

bool cmp(long a, long b)
{
    if (a > b)
        return true;
    return false;
}

// Cách 2: xét qua vector thay vì sài array
```

Sort_STL_NhieuTieuChi.cpp

STL_09 > Sort_STL_NhieuTieuChi.cpp > ...

Click here to ask Blackbox to help you code faster

```
/*
    C++ Buổi 10_Bài 02.Sắp xếp theo nhiều tiêu chí.
    Cho mảng số nguyên A[] có N phần tử, hãy sắp xếp các phần tử trong mảng theo các tiêu trí sau đây:

    1.Sắp xếp các phần tử theo giá trị tuyệt đối tăng dần.
    2.Sắp xếp các phần tử theo giá trị tuyệt đối tăng dần.Nếu 2 số có cùng giá trị tuyệt đối thì số dương được xếp sau.
    3.Sắp xếp theo tổng chữ số tăng dần.
    4.Sắp xếp theo tổng chữ số tăng dần.Nếu 2 số có cùng tổng chữ số thì in ra số có giá trị nhỏ hơn sẽ xếp sau.
    5.Sắp xếp sao cho các số chẵn xếp trước,các số lẻ xếp sau.
    6.Sắp xếp sao cho các số chẵn xếp trước theo thứ tự giảm dần,các số lẻ xếp sau theo thứ tự tăng dần.

    Dữ liệu vào:
    Dòng đầu tiên là số nguyên dương N.
    Dòng thứ 2 là N phần tử trong mảng, các phần tử viết cách nhau một dấu cách. (1<=N<=10^5;-10^9<=A[i]<=10^9)

    Dữ liệu ra:
    In ra đáp án của mỗi yêu cầu của bài toán trên 1 dòng.

    Dữ liệu vào:
    5
    14 -88 6 23 -14
    Dữ liệu ra:
    6 14 -14 23 -88
    6 -14 14 23 -88
    -88 -14 14 23 6
    -88 -14 23 14 6
    -88 -14 14 6 23
    14 6 -14 -88 23
*/
```

```cpp
#include <iostream>
#include <algorithm>
using namespace std;

void NhapMang(long[], int&);
bool cmp1(long, long);
bool cmp2(long, long);
int TongChuSo(long);
bool cmp3(long, long);
bool cmp4(long, long);
bool checkChan(long);
bool checkLe(long);
bool cmp5(long, long);
bool cmp6(long, long);
void XuatMang(long[], int);

int main()
{
    int n;
    cin >> n;
    long a[1000001];
    NhapMang(a, n);
    XuatMang(a, n);
    return 0;
}
```

```
57  void NhapMang(long a[], int& n)
58  {
59      for (int i = 0; i < n; i++)
60          cin >> a[i];
61  }
62
63  bool cmp1(long a, long b)
64  {
65      if (abs(a) < abs(b))
66          return true;
67      return false;
68  }
69
70  bool cmp2(long a, long b)
71  {
72      if (abs(a) != abs(b))
73      {
74          if (abs(a) < abs(b))
75              return true;
76          return false;
77      }
78      else
79      {
80          if (a < b)
81              return true;
82          return false;
83      }
84  }
85
```

```
86  int TongChuSo(long n)
87  {
88      int sum = 0;
89      while (abs(n) != 0)
90      {
91          if (abs(n) >= 10)
92              sum += abs(n) % 10;
93          else
94          {
95              // trường hợp chỉ có 1 chữ số
96              // thì n: âm => cộng giá trị âm
97              // n: dương => cộng giá trị dương
98              sum += n;
99          }
100         n /= 10;
101     }
102     return sum;
103 }
104
105 bool cmp3(long a, long b)
106 {
107     if (TongChuSo(a) < TongChuSo(b))
108         return true;
109     return false;
110 }
```

```
112  bool cmp4(long a, long b)
113  {
114      if (TongChuSo(a) != TongChuSo(b))
115      {
116          if (TongChuSo(a) < TongChuSo(b))
117              return true;
118          return false;
119      }
120      else
121      {
122          if (a > b)
123              return true;
124          return false;
125      }
126  }
127
128  bool checkChan(long n)
129  {
130      return (n % 2 == 0);
131  }
132
133  bool checkLe(long n)
134  {
135      return (n % 2 != 0);
136  }
137
138  bool cmp5(long a, long b)
139  {
140      if (checkChan(a) && checkLe(b))
141          return true;
142      return false;
143  }
```

```
145   bool cmp6(long a, long b)
146   {
147       if (checkChan(a) && checkChan(b))
148       {
149           if (a > b)
150               return true;
151           return false;
152       }
153       else if (checkLe(a) && checkLe(b))
154       {
155           if (a < b)
156               return true;
157           return false;
158       }
159       else if (checkChan(a) && checkLe(b))
160           return true;
161       return false;
162   }
```

```cpp
void XuatMang(long a[], int n)
{
    // có thể sử dụng for each thay cho for thông thường
    // tiêu chí 1
    sort(a + 0, a + n, cmp1);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;

    // tiêu chí 2
    sort(a + 0, a + n, cmp2);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;

    // tiêu chí 3
    sort(a + 0, a + n, cmp3);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;

    // tiêu chí 4
    sort(a + 0, a + n, cmp4);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;

    // tiêu chí 5
    sort(a + 0, a + n, cmp5);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;

    // tiêu chí 6
    sort(a + 0, a + n, cmp6);
    for (int i = 0; i < n; i++)
        cout << a[i] << " ";
    cout << endl;
}
```