Nguồn

CNTT1 - K62: + Trịnh Thành Nam + Trần Công Chiến + Lê Hữu Chung + Phạm Việt Hùng	

Cấu trúc dữ liệu

- Theme: Light (Visual Studio - C\C++)

Stack

Kiểm Tra Ngoặc Đúng	Code

```
#include<bits/stdc++.h>
using namespace std;
/* Ý tưởng : dùng stack để lưu các dấu đóng ngoặc
Duyệt qua chuỗi ngoặc nếu gặp dấu mở ngoặc (thỏa mãn yêu cầu bài toán) thì đẩy dấu đóng
ngoặc tương ứng vào stack nếu gặp dấu đóng ngoặc thì kiểm tra xem dấu ngoặc mới nhất đẩy vào
stack có trùng với dấu ngoặc hiện tại không nếu có thì xóa phần tử mới nhất trong stack đi
Cuối cùng kiểm tra xem stack rỗng và chưa gặp 1 trường hợp nào không thỏa mãn bài toán thì
là chuỗi ngoặc đúng (sai khi 1 trong 2 điều kiện trên sai)*/

int main()
{
    map <char, char> mp{{'(', ')'}, {'[', ']'}, {'{', '}'}}; // dùng để khi gặp dấu mở ngoặc
chúng ta sẽ đẩy dấu đóng ngoặc vào stack
    map <char, int> ut{{'(', 1}, {'[', 2}, {'{', 3}, {')', 1}, {']', 2}, {'}', 3}}; // dùng
để lưu độ ưu tiên của các dấu ngoặc
    int t; cin >> t;
    while(t--)
    {
        stack<char> s;
        string x;
        cin >> x;
        int check = 1;
        for(char c : x){ // duyệt qua từng kí tự
            if(c == '(' || c == '[' || c == '{') {
                if(s.size() && ut[c] > ut[s.top()]) { // Kiểm tra nếu stack không rỗng và độ
ưu tiên của kí tự hiện tại lớn hơn kí tự mới nhất được đẩy vào stack -> sai yêu cầu
                                                    // bài toán nên cho check = 0 và thoát
khỏi vòng lặp
                    check = 0;
                    break;
                }
                s.push(mp[c]); // đẩy các dấu đóng ngoặc vào stack
            }
            else
            {
                if(s.empty() || s.top() != c) // nếu stack rỗng mà gặp dấu đóng ngoặc hoặc
phần tử đầu trong stack không trung với kí tự hiện tại -> chuỗi ngoặc sai
                {
                    check = 0;
                    break;
                }
                else s.pop(); // nếu trùng thì xóa phần tử đầu tiện trong stack
            }
        }
        if(check == 1 && s.empty()) cout << "Dung" << endl;
        else cout << "Sai" << endl;
    }
}
```

Khối lượng hóa chất	Code

```
#include<bits/stdc++.h>
using namespace std;
/* Đề bài : https://www.laptrinhonline.club/problem/tichpxweighthoachat

Một chất hữu cơ chỉ gồm các nguyên tố hóa học C,H,O có khối lượng phân tử tương ứng C=12,
H=1, O=16.
Ví dụ Axit capric có công thức CH3-CH2-CH2-CH2-CH2-CH2-CH2-CH2-CH2-COOH ở dạng rút gọn
CH3(CH2)8COOH
Bài toán đặt ra là cho một công thức ở dạng rút gọn bạn hãy tính khối lượng hóa chất đó biết
rằng số rút gọn chỉ nằm trong khoảng [2,9].*/

/*Ý tưởng : Dùng map để lưu khối lượng phân tử của các nguyên tố hóa học và dấu ( (quy ước
là 0)
          Dùng stack để lưu giá trị*/


int main()
{
```

```
    int test;
    cin >> test;
    while(test--)
    {
        map<char,int> K = {{'C', 12}, {'O', 16}, {'(', 0}, {'H', 1}}; // quy ước ( = 0
        string x;
        cin >> x;
        stack<int> S;
        for(auto c : x)
        {
            if(K.find(c) != K.end()) S.push(K[c]); // Nếu kí tự là COH hoặc ( thì đẩy vào
stack
            else if(c == ')') // tính tổng trong ngoặc
            {
                int t = 0;
                while(S.top() != 0) // cộng vào chừng nào gặp dấu ( thì dừng (quy ước là 0)
                {
                    t += S.top();
                    S.pop();
                }
                S.top() = t; // dấu ( vẫn còn trong stack -> thay giá trị của dấu ( bằng giá
trị vừa tính được
            }
            else S.top() *= c-'0';// Nếu là 1 số thì nhân phần tử đầu tiên của stack với số
đó
        }
        int t = 0;
        while(S.size()) // cộng tất cả các giá trị trong stack lại -> tổng
        {
            t += S.top();
            S.pop();
        }
        cout << t << "\n";
    }
}
```

Chào đón tân sinh viên K59	Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n; cin >> n;
    // a[i]: chiều cao người thứ i
    // L[i]: vị trí của người gần nhất bên trái cao hơn người thứ i
    // Nếu L[i] = -1, nghĩa là ở bên trái không có ai cao hơn người i

    // R[i]: tương tự L[i], nhưng là về bên phải
    int a[n], L[n], R[n];

    for (auto &x : a) cin >> x; // Nhập dữ liệu cho dãy a

    // SL: container tạm thời dùng khi duyệt dãy a từ trái -> phải, SR: phải -> trái
    stack <pair <int, int>> SL, SR; // Mỗi phần tử có dạng {vị trí, chiều cao}

    SL.push({-1, INT_MAX});
    //Duyệt từ trái -> phải
    for(int i = 0; i < n; i++)
    {
        //Khi chiều cao của người ở đỉnh stack <= của người thứ i
        // thì bỏ người ở đỉnh stack ra
        while(SL.top().second <= a[i]) SL.pop();

        // L[i] = vị trí của người ở đỉnh stack
        L[i] = SL.top().first;

        // Đưa người thứ i vào stack
        SL.push({i, a[i]});
    }
    // -> Duyệt từ trái -> phải ta thu được mảng L[]
    // ==> Duyệt trừ phải -> trái ta sẽ thu được mảng R[]

    SR.push({-1, INT_MAX});
    //Duyệt từ phải -> trái
    for(int i = n - 1; i >= 0; i--)
    {
        while(SR.top().second <= a[i]) SR.pop();
        R[i] = SR.top().first;
        SR.push({i, a[i]});
    }

    for(int i = 0; i < n; i++)
        // Nếu 1 trong 2 giá trị (L[i], R[i]) bằng -1 thì in ra giá trị còn lại
```

```
        // Cộng thêm 1 để triệt tiêu đi giá trị -1 trong tổng (L[i] + R[i])
        if(L[i] == -1 || R[i] == -1)
            cout << L[i] + R[i] + 1 << " ";
        else
            // khoảng cách từ (i -> L[i]) <= (i -> R[i]) thì in ra L[i], ngược lại in
R[i]
            cout << (i - L[i]) <= (R[i] - i) ? L[i] : R[i] << " ";
}
```

Biểu thức hậu tố Ba Lan	Code

```
/*Biểu thức số học mà các toán tử chỉ gồm các phép tính + , -, *, / trên trường số nguyên và
các toán hạng là các số tự nhiên nằm trong đoạn [0,9].
Bạn hãy lập trình chuyển đổi sang dạng hậu tố Ba Lan sau đó tính giá trị của biểu thức.*/

/*ý tưởng: tạo 1 stack để chứa các dấu (,+,-,*,/
cho dữ liệu in chạy:
 + nếu là 0->9 thì thêm vào out
 + nếu là '(' thì cất vào stack đến khi gặp ')' sẽ thêm các dấu đã lưu trong stack vào out
và pop toàn bộ
 + nếu là +,-,*,/ thì thêm vào out các dấu đã lưu trong stack đến hết, và thêm dấu đó vào
stack
cuối cùng thêm lại toàn bộ stack vào out */
#include<bits/stdc++.h>
using namespace std;
map <char, int> UT = {{'(', 0}, {'+', 1}, {'-', 1}, {'*', 2}, {'/', 2}};
int kq(int a,int b,char c)
{
    if(c == '+') return a + b;
    if(c == '-') return a - b;
    if(c == '*') return a * b;
    return a/b;
}
int giatri(string out)
{
    stack<int> S;
    for(char c:out)
    if('0' <= c && c <= '9') S.push(c - '0');
    else
    {
        int a = S.top(); S.pop();
        int b = S.top(); S.pop();
        S.push(kq(b, a, c));
    }
    return S.top();
}

string Balan(string in)
{
    string out = "";
    stack<char> S;
    for(char c : in)
    if('0' <= c && c <= '9')
        out += c;
    else if(c == '(') S.push(c);
    else if(c == ')')
    {
        while(S.top() != '(')
        {
            out += S.top();
            S.pop();
        }
        S.pop(); //lấy nốt dấu '(' ra khỏi stack
    }
    else //toán tử
    {
        while(S.size() && UT[S.top()] >= UT[c])
        {
            out += S.top();
            S.pop();
        }
        S.push(c);
    }
    while(S.size())
    {
        out += S.top();
        S.pop();
    }
    return out;
}
int main()
{
    string in, out = "";
    cin >> in;
```

```
    out = Balan(in);
    cout << out << "\n" << giatri(out);
}
```

Cài đặt Stack bằng mảng	Code

```
#include<bits/stdc++.h>
using namespace std;

template <class T>
class Stack
{
    int n,cap;  //n-size, cap-capacity
    T *buf;     //buffer chứa các phần tử của stack
    public:
        Stack()
        {
            n = cap = 0;
            buf = NULL;
        }

        ~Stack() {if(buf) delete []buf;}
        bool empty() {return n == 0;}
        int size() {return n;}
        void pop() {n--;}

        T &top() //read-write function
        {
            return buf[n-1];
        }

        void push(T x)
        {
            if(n == cap) // full
            {
                cap = 1 + cap * 2;   //cap == 0 ? 1 : cap * 2;
                T *temp = new T [cap];
                for(int i = 0; i < n; i++)
                    temp[i] = buf[i]; //copy tất cả phần tử của buf sang temp
                if(buf) delete []buf;
                buf = temp;
            }
            buf[n++] = x;
        }
};
```

Cài đặt Stack bằng linked list	Code

```
#include<bits/stdc++.h>
using namespace std;
// #ifndef ___stack__cpp___
// #define ___stack__cpp___
template <class T>
struct Node
{
    T elem;
    Node *next;
    Node(T e, Node*N = 0) //hàm tạo 1 Node
    {
        elem = e;
        next = N;
    }
};

template <class T>
class STACK
{
    Node<T> *Head;  //Con trỏ trỏ vào top
    int n;          //size
    public:
        STACK()
        {
            Head = NULL;
            n = 0;
        }

        ~STACK(){while(n) pop();}
        int size(){return n;}
        bool empty(){return n == 0;}
        void push(T x)
        {
            Head = new Node<T> (x,Head);
            n++;
        }
```

```
        void pop()
        {
            Node<T> *p = Head;
            Head = Head->next;
            delete p;
            n--;
        }

        T &top(){return Head->elem;}
};
// #endif
int main()
{
    STACK<int> *S = new STACK <int>();
    for(int x : {234, 62, 27, 63, 724, 846})
        S->push(x);
    S->top() = 0;
    while(S->size())
    {
        cout << S->top() << " ";
        S->pop();
    }
    delete S;
}
```

Queue

Giới thiệu queue	Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    queue<int> Q;
    for (int x:{4, 7, 2, 8, 1, 6})
        Q.push(x);
    Q.front() = 5;
    Q.back() = 9;
    while(not Q.empty())
    {
        cout<<Q.front()<<" ";
        Q.pop();
    }
}
```

Bịt mắt bắt dê	Code

```
//Bịt mắt bắt dê
/* Bài toán đặt ra: có n người đánh số từ 1 -> n, bắt đầu từ người số 1 cứ đếm đến người
thứ k thì loại ra khỏi vòng tròn và bắt đầu chơi tiếp từ người thứ k+1, được đếm từ 1 cứ
tiếp tục như vậy vì đứng thành vòng tròn nên lần lượt sẽ loại hết chỉ còn người cuối
cùng. Hãy cho biết chỉ số của người cuối cùng là bao nhiêu?*/

/*Ý tưởng: Tạo queue Q lưu chỉ số người chơi ta chạy chỉ số đến người thứ k thì loại k
nếu không phải k thì push lại vào Q*/
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, k; cin >> n >> k;
    queue <int> Q;
    //cất chỉ số vào Q
    for(int i = 1; i <= n; i++)
        Q.push(i);

    while(Q.size() > 1)
    {
        //chạy chỉ số đến người thứ k thì loại không thì push lại vào Q
        for(int i = 1; i < k; i++)
        {
            Q.push(Q.front());
            Q.pop();
        }
        Q.pop();
    }
    cout << Q.front();
}
```

Tập tam giác	Code

```cpp
//Cho dãy số nguyên dương a1,a2,…,an từ dãy đã cho hãy chọn các phần tử
//tùy ý mỗi phần tử không quá 1 lần được 1 dãy mới nhiều phần tử nhất có thể
//sao cho lấy ra 3 giá trị bất kỳ có chỉ số khác nhau đều tạo thành 3 cạnh của tam giác.

/*
Ý tưởng: tạo ra 1 queue Q lưu bộ cạnh của 1 tam giác thỏa mãn:
sắp xếp theo thứ tự giảm dần
nếu Q còn phần tử và x+Q.back<=Q.front thì xóa Q (tức là tổng 2 cạnh nhỏ hơn hoặc bằng cạnh
còn lại)
so sánh res và Q.size: res=res>Q.size()?res:Q.size()
lưu ý: do ta đã sắp xếp giảm dần nên các số bằng nhau được xét hết 1 lúc
nếu res<=2 tức chỉ có 2 hoặc 1 cạnh nên không thành 1 tam giác
*/
#include<bits/stdc++.h>
using namespace std;
int main()
{
    queue<int> Q;
    int n, res =0;
    cin >> n;
    int a[n];
    for(auto &x:a) cin>>x;   //cin a[0]...a[n-1]
    sort(a, a + n, greater<int>()); //sx giảm dần
    for(int x:a)
    {
        while(Q.size() && x + Q.back() <= Q.front())
            Q.pop();
        Q.push(x);
        if(res < Q.size()) res = Q.size();
    }
    if(res > 2) cout << res;
    else cout<<"Khong the tao ra day thoa man";
}
```

Búp bê Nga	Code

```cpp
/*Đề bài:Có n con búp bê Nga có thể lồng vào nhau có thể lồng vào nhau nếu chênh lệch kích
thước là k hoặc nhỏ hơn.
Xuất ra số búp bê ngoài cùng và tổng kích thước số búp bê ngoài cùng đó.*/
/*Ý tưởng: sắp xếp các búp bê theo kích thước giảm dần
Tạo 1 queue Q để chứa những búp bê bị lồng trong cùng,
Nếu 1 con búp bê có thể bị lồng thì sẽ pop con lồng nó nếu không bị lồng sẽ là con ngoài
cùng. */
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, k; cin >> n >> k;
    int a[n];
    for(int &x : a) cin >> x;  //Nhập a[0]...a[n-1]
    sort(a, a + n, greater<int>()); //sắp xếp giảm dần
    queue<int> Q;
    int res = 0;  //chứa tổng kích thước của những con ngoài cùng
    for(auto x : a)
    {
        Q.push(x);
        if(Q.front() >= x + k)
            Q.pop();
        else res += x;
    }
    cout << Q.size() << " " << res;
}
```

Cánh cửa thần kỳ	Code

```cpp
/*Vì sau khi qua cửa sẽ gấp đôi chỉ số ta quy bài thành dãy 1234511223344551111222... tìm số
thứ n trong dãy.
Ý tưởng:tạo 1 cấu trúc map hoặc mảng để lưu chỉ số ứng với tên người
tạo 1 queue q chứa chỉ số và số chỉ số trong dãy.*/
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int id = 1;
    map< int, string> mp;
    for (string i : {"dangdungcntt", "tienquanutc", "quang123", "maianh",
"nguyenminhduc2820"})
        mp[id++] = i;

    int t; cin >> t;
    while(t--)
    {
        int n; cin >> n;
        queue<pair<int, int>> q;
```

```
        for(int x : {1,2,3,4,5})
            q.push({x, 1});

        while(n > q.front().second)
        {
            n -= q.front().second;
            q.push(q.front());
            q.back().second*=2;
            q.pop();
        }

        cout << mp[q.front().first] << "\n";
    }
}
```

Cài đặt queue bằng mảng	Code

```
#include<bits/stdc++.h>
using namespace std;
template <class T>
class QUEUE
{
    int n, F, L, cap;
    T *buf;
    public:
        QUEUE() {buf = NULL; n = 0; F = L = 0;}
        ~QUEUE(){if(buf) delete [] buf;}
        int size(){return n;}
        bool empty(){return n == 0;}
        T&front() {return buf[F];}
        T&back() {return L == 0? buf[cap-1] : buf[L-1];}
        void pop() {n--; if(n == 0) F = L = 0; else F = (F+1)%cap;}
        void push(T x)
        {
            if(n == cap) //mở rộng bộ nhớ
            {
                cap = cap ? cap*2 : 1;
                T*tem = new T[cap];
                for(int i = F, j = 0; i < F + n; i++, j++)
                    tem[j] = buf[i % cap];
                if(buf) delete[]buf;
                buf = tem;
                F = 0;
                L = n;
            }
            buf[L] = x; L = (L + 1)%cap;
            n++;
        }
};
int main()
{
    QUEUE<int> Q;
    for(int x : {53, 526, 21, 1364, 236})
        Q.push(x);
    Q.front() = 100;
    Q.back() = 20;
    while(Q.size())
    {
        cout << Q.front() << " ";
        Q.pop();
    }
}
```

Cài đặt queue bằng linked list	Code

```
#include<bits/stdc++.h>
using namespace std;
template <class T>
struct node
{
    T elem;
    node *next;
    node(T x) {elem = x; next = NULL;}  //hàm tạo
};
template <class T>
class Queue
{
    node<T> *Head,*Tail;
    int n;
    public:
        Queue() {Head = Tail = NULL; n = 0;}
        ~Queue()
        {
            while(Head != NULL)
```

```
            {
                node<T>*p = Head;
                Head = Head->next;
                delete p;
            }
        }
        int size() {return n;}
        bool empty() {return n == 0;}
        T &front() {return Head->elem;}
        T &back() {return Tail->elem;}
        void push(T x)
        {
            // Cách 1
//          if(n>0)
//          {
//              node<T> *p = new node<T>(x);
//              Tail->next = p;
//              Tail = p;
//          }
//          else
//          {
//              Head = new node<T>(x);
//              Tail = Head;
//          }

            // Cách 2
            if(n == 0) Head = Tail = new node<T>(x);
            else Tail = Tail->next = new node<T>(x);
            n++;
        }

        void pop()
        {
            node<T>*p = Head;
            Head = Head->next;
            delete p;
            n--;
        }
};

int main()
{
    Queue<int> q;
    for (int i : {3, 2, 4, 9, 0, -12, 223})
        q.push(i);
    while (q.size())
    {
        cout << q.front() << " ";
        q.pop();
    }

}
```

Vector

Giới thiệu Vector	Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    vector <int> A;  //Vector không có phần tử
    cout << "\nsize : " << A.size();
    cout << "\ncapacity : " << A.capacity();
    vector<int> B(5,1);  //Vector chứa 1 1 1 1 1
    B.push_back(3);
    cout << "\nsize : " << B.size();
    cout << "\ncapacity : " << B.capacity();

    cout << "\nB: ";
    for(int i = 0; i < B.size(); i++)

    cout << B[i] << " ";
    B.front() = 4;
    B.back() = 6;
    B[2] = 2;
    B.at(3) = 5;

    cout << "\nB: ";
    for(int x : B)
        cout << x << " ";

    cout << "\nDuyet xuoi B: ";

    for (vector<int>::iterator it = B.begin(); it != B.end(); it++)
        cout << *it << " ";
```

```
    cout << "\nDuyet nguoc B: ";
    for(vector<int>::reverse_iterator it = B.rbegin(); it != B.rend(); it++)
        cout << *it << " ";
}
```

Cài đặt Vector bằng mảng	Code

```
#include<bits/stdc++.h>
using namespace std;
template<class T>
class vt_rite //bộ lặp ngược
{
    T*curr;
    public:
    vt_rite(T *c = NULL){curr = c;}
    vt_rite<T> &operator = (vt_rite<T> it)
    {
        this->curr = it.curr;
        return *this;
    }

    bool operator != (vt_rite<T> it){return this->curr != it.curr;}
    T &operator *(){return *curr;}

    vt_rite<T> operator++(int) //it++
    {
        T* c = curr;
        curr = curr-1;
        return vt_rite<T>(c);
    }

    vt_rite<T> operator++() //++it
    {
        curr = curr-1;
        return vt_rite<T>(curr);
    }
};

template<class T>
class Vector
{
    int n, cap;  //n-size, cap-capacity
    T *buf;     //luu cac phan tu
    private:
        void recap(int k)  //mo rong kha nang luu theo k
        {
            if(cap >= k) return;
            cap = k;
            T *tem = new T[cap];
            for(int i = 0; i < n; i++)
                tem[i] = buf[i];
            if(buf) delete[]buf;
            buf = tem;
        }

    public:
        typedef T*iterator;
        iterator begin() {return buf;}
        iterator end() {return buf + n;}
        Vector()
        {
            n = cap = 0;
            buf = NULL;
        }

        Vector(int k, T x) //
        {
            n = cap = k;
            buf = new T[k];
            for(int i = 0; i < k; i++)
                buf[i] = x;
        }

        ~Vector() {if(buf) delete[]buf;}
        int size() {return n;}
        bool empty(){return n == 0;}
        T &front() {return buf[0];}
        T &back()  {return buf[n - 1];}
        T &operator[](int i) {return buf[i];}
        T &at(int i) {return buf[i];}
        void pop_back() {n--;}
```

```
    void push_back(T x)
    {
        if(n == cap) recap(cap ? cap*2 : 1);
        buf[n++] = x;
    }
    void resize(int k, T x)
    {
        if(n >= k)
        {
            n = k;
            return;
        }
        if(k > cap) recap(k);
        for(int i = n; i < k; i++) buf[i] = x;
        n = k;
    }

    void insert(iterator it, T x)
    {
        if(n == cap)
        {
            int k = it - buf;
            recap(cap ? cap*2 : 1);
            it = buf + k;
        }
        for(iterator it1 = buf + n - 1; it1 >= it; it1--)
            *(it1 + 1) = *it1;
        *it = x;
        n++;
    }
};
int main()
{
    Vector<int> A(5,3);
    A.front() = 7;
    A.back() = 4;
    for(auto x : A)
        cout << x << " ";

    cout << endl;
    for(int i = 0; i < A.size(); i++)
        cout << A[i] << " ";

    cout << endl;
    for(Vector<int>::iterator it = A.begin(); it != A.end(); it++)
        cout << *it << " ";
}
```

Tính giai thừa	Code

```
//Tính n! (1<=n<=1000)
//Ý tưởng:ta sẽ tính giai thừa xong lưu vào vector A nhưng lưu ngược
//để nếu tăng thêm 1 chữ số sẽ chỉ cần push_back vào A sẽ dễ hơn.

#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n; cin >> n;
    vector<int> A(1,1);   //tạo vector 1
    for(int i = 2; i <= n; i++){
        long long nho = 0;
        for(auto &x:A)
        {
            nho += x*i;
            x = nho%10;
            nho/=10;
        }
        while(nho) {A.push_back(nho % 10); nho /= 10;}
    }
    reverse(A.begin(), A.end());
    for(auto x:A) cout << x;
}
```

Tính số fibonacci lớn	Code

```
//Số Fibonacci lớn
// Đề bài: tìm số Fibonacci thứ n (1<=n<=1000)

// Ý tưởng: f[1] = 1, f[2] = 1, f[n] = f[n-1] + f[n-2]. Ta dùng vector để lưu số thay vì
dùng biến vì số lượng lưu sẽ nhiều hơn.
```

```cpp
// Lưu f[n-2] bằng vector a, f[n-1] bằng vector b, f[n] bằng vector c.
#include<bits/stdc++.h>
using namespace std;
vector<int> cong(vector<int> a,vector<int> b)
{
    vector<int> c = a;
    if(a.size() < b.size()) c.resize(b.size(), 0);
    for(int i = 0; i < b.size(); i++)
        c[i] += b[i];
    int nho = 0;
    for(auto &x:c)
    {
        nho += x;
        x = nho%10;
        nho /= 10;
    }
    if(nho) c.push_back(nho);
    return c;
}
int main(){
    int n;
    cin >> n;
    if (n == 0 || n == 1) {cout << n; return 0;}
    vector <int> a(1,0), b(1,1), c;
    for(int i = 2; i <= n; i++)
    {
        c = cong(a,b);  //a, b truyền vào thông qua toán tử copy, c= toán tử gán
        a = b;
        b = c;
    }
    for(auto it = b.rbegin(); it != b.rend(); it++) cout << *it;
}
```

List

Giới thiệu List	Code

```cpp
#include<bits/stdc++.h>
#include<list>
using namespace std;
int main()
{
    list<int> L(5, 3);
    L.front() = 7; L.back() = 8;   //7 3 3 3 8
    L.push_front(1); L.push_back(6);  //1 7 3 3 3 8 6

    cout << "L: ";
    for(list<int>::iterator it = L.begin(); it != L.end(); it++)
        cout << *it << " ";

    L.pop_front();
    L.pop_back();
    cout << "\n\nXoa dau va cuoi\nL: ";
    for(list<int>::iterator it = L.begin(); it != L.end(); it++)
        cout << *it << " ";


    cout << "\n\nTruoc sort L : ";
    for(auto x : L)
        cout << x << " ";
    L.sort();
    cout << "\nSau sort L : ";
    for(auto x : L)
        cout << x << " ";

    auto it = L.begin(); it++; it++; ++it;
    L.insert(it, 9);
    cout << "\n\nChen 9 vao vi tri 3\nL : ";
    for(auto x : L)
        cout << x << " ";

    it = L.begin(); it++; it++;
    L.erase(it);  //xoa so 3

    it = L.begin();
    while(*it != 7) it++;
    L.erase(it);
    cout << "\n\nXoa di so o vi tri so 2 va so co gia tri = 7";

    cout << "\nL : ";
    for(auto x : L)
        cout << x << " ";
    cout << "\nsize : " << L.size();
}
```

Xóa k chữ số được số lớn nhất	Code

```
/*Ý tưởng: ta xét những chữ số nào nhỏ nhất trong đoạn k để tạo ra số lớn nhất.
Tạo 1 list L để chứa những chữ số lớn nếu chữ số tiếp theo mà lớn hơn trước thì pop_back và
cuối cùng thêm kí tự*/
#include<bits/stdc++.h>
using namespace std;
void xoa(string x,int k)
{
    list<char> L;
    for(auto c:x)
    {
        while(L.size() && k > 0 && L.back() < c)
            {
                L.pop_back();
                k--;
            }
        L.push_back(c);
    }

    while(k > 0)
    {
        L.pop_back();
        k--;
    }
    for(auto z : L)
        cout << z;
    cout << "\n";
}
int main()
{
    string x;
    int test, k;
    cin >> x >> test;
    while(test--)
    {
        cin >> k;
        xoa(x, k);
    }
}
```

Trinh thám	Code

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, k, x;
    list <pair<int, int>>L;
    cin >> n >> k;
    for(int i = 1; i <= n; i++)
    {
        cin >> x;
        while(L.size() && L.back().second <= x)
            L.pop_back();
        L.push_back({i, x});
        if(i >= k)
        {
            while(i - L.front().first >= k)
                L.pop_front();
            cout << L.front().second << " ";
        }
    }
}
```

Danh sách liên kết

Danh sách liên kết đơn	Code

```
#include<bits/stdc++.h>
using namespace std;
//Khai báo một node
template<class T>
struct Node
{
    T elem;
    Node *next;
    Node(T e,Node *N=NULL) {elem = e; next = N;}
};
template <class T>
class slist_ite  //Bộ lặp xuôi
{
    Node<T> *curr;
```

```
    public:
        Node<T>*&getcur(){return curr;}
        slist_ite(Node<T>*c=NULL) {curr=c;}  //hàm tạo
        slist_ite<T>& operator=(slist_ite<T> const &it)  //toán tử gán
        {
            this->curr=it.curr;
            return *this;
        }
        bool operator!=(slist_ite<T> it)  //so sánh !=
        {
            return curr!=it.curr;
        }
        T &operator*(){return curr->elem;}  //toán tử lấy giá trị *it
        slist_ite<T> operator++()  //++it
        {
            curr=curr->next;
            return curr;
        }
        slist_ite<T> operator++(int)  //it++
        {
            Node<T>*p=curr;
            curr=curr->next;
            return p;
        }
};
template<class T>
class slist                 //danh sách liên kết đơn
{
    Node<T> *Head,*Tail;    //Hai con trỏ trỏ vào đầu và cuối
    int n;                   //size
    public:
        slist() {Head = Tail = NULL; n = 0;}
        ~slist()
        {
            if(n > 0)
            {
                for(Node<T>*p = Head->next; p != NULL; p = p->next)
                {
                    delete Head;
                    Head=p;
                }
                delete Head;
                n = 0;
                Head = Tail = 0;
            }
        }
        int size() {return n;}
        bool empty() {return n == 0;}
        T &front() {return Head->elem;}
        T &back() {return Tail->elem;}
        T &operator[](int k)              //list trong STL không có hàm này
        {
            Node<T> *p = Head;
            while(k--) p = p->next;
            return p->elem;
        }
        void push_front(T x)
        {
            if(n == 0) Head = Tail = new Node<T>(x);
            else Head = new Node<T>(x,Head);
            n++;
        }
        void push_back(T x)
        {
            if(n == 0) Head = Tail = new Node<T>(x);
            else Tail = Tail->next = new Node<T>(x);
            n++;
        }
        void pop_back()
        {
            if(n == 1) {delete[]Head; Head = Tail = NULL;}
            else
            {
                Node<T>*p = Head;
                while(p->next  != Tail) p = p->next;
                delete Tail;
                Tail = p;
                Tail->next = NULL;
            }
            n--;
        }
        void pop_front()
        {
            if(n == 1) {delete[]Head; Head = Tail = NULL;}
            else
            {
```

```
            Node<T>*p = Head; Head = Head->next; delete p;
        }
        n--;
    }
    typedef slist_ite<T> iterator;
    iterator begin(){return Head;}
    iterator end() {return NULL;}
    void insert(iterator it,T x)
    {
        Node<T>*q=it.getcur();
        Node<T>*p=new Node<T>(q->elem,q->next);
        q->next=p;
        q->elem=x;
        if(q==Tail) Tail=p;
        n++;
    }
    void erase(iterator it)
    {
        Node<T>*p=it.getcur();
        if(p==Head) pop_front();
        else if(p==Tail) pop_back();
        else
        {
            Node<T>*q = Head; while(q->next != p) q=q->next;
            q->next = p->next;
            delete p;
            n--;
        }
    }
    void sort(bool ok = true) //mặc định sắp xếp tăng dần
    {
        if(n<=1) return;
        for(Node<T>*p = Head; p != NULL; p = p->next)
        for(Node<T>*q = p->next; q != NULL; q = q->next)
        if(q->elem < p->elem == ok) swap(p->elem, q->elem);
        //đổi mối nối thay vì đổi dữ liệu
    }
};

int main()
{
    slist<int> L;
    for(int x:{2, 7, -8, 12, 0, -21, 43, 5, 6, 12})
        x % 2 == 0?L.push_front(x) : L.push_back(x);

    cout << "\nL: ";
    for(slist<int>::iterator it=L.begin();it!=L.end();it++)
        cout<<*it<<" ";

    slist<int>::iterator it1;
    it1 = L.begin();
    L.insert(it1, -5);
    cout << "\nL: ";
    for(auto z:L) cout << z << " ";

    it1 = L.begin();
    for(int i = 1; i <= 5; i++) it1++;
    L.insert(it1, -3);
    cout<<"\nL: ";
    for(auto z:L) cout << z << " ";

    it1 = L.begin();
    for(int i = 1; i < L.size(); i++) it1++;
    cout << "\n" << *it1 << "\n";

    L.insert(it1,-7);
    cout<<"\nL: ";
    for(auto z:L) cout << z << " ";
    L.push_back(100);
    cout<<"\nL: ";
    for(auto z:L) cout << z << " ";

    L.erase(L.begin());
    it1 = L.begin();
    for(int i = 1; i < L.size(); i++) it1++;
    L.erase(it1);
    it1 = L.begin();
    for(int i = 1; i <= 7; i++) it1++;
    L.erase(it1);
    cout<<"\nL: ";
    for(auto z:L) cout<<z<<" ";

    L.sort(false);
    cout<<"\nL: "; for(auto z : L) cout<<z<<" ";
}
```

Danh sách liên kết kép	Code

```
#include<bits/stdc++.h>
using namespace std;
template <class T>
struct node
{
    T elem;//element
    node *prev,*next; //prev : con trỏ quản lý node phía trước, next con trỏ quản lý node
phía sau
    node(T e,node<T> *P = 0, node<T> *N = 0) //Hàm tạo
    {
        elem = e;
        prev = P;
        next = N;
    }
};
template <class T>
class dlist_ite  //Bộ lặp xuôi
{
    node<T> *curr;
    public:
        node<T>*&getcur(){return curr;} //Trả về con trỏ curr - có thể vừa ghi vừa đọc
        dlist_ite(node<T>*c = NULL) {curr = c;}  //Hàm tạo
        dlist_ite<T>& operator=(dlist_ite<T> const &it)  //Toán tử gán
        {
            this->curr = it.curr;
            return *this;
        }
        bool operator!=(dlist_ite<T> it)  //Toán tử khác - so sánh không bằng
        {
            return curr != it.curr;
        }
        T &operator*(){return curr->elem;}  //Toán tử lấy giá trị *it - có thể vừa ghi vừa
đọc
        dlist_ite<T> operator++()  //++it
        {
            curr=curr->next;//Gán curr là node tiếp theo
            return curr;//Trả về curr
        }
        dlist_ite<T> operator++(int) //it++
        {
            node<T>*p = curr;//Tạo node p=curr
            curr = curr->next;//Gán curr là node tiếp theo
            return p;//Trả vè p
        }
};
template <class T>
class dlist_rite  //Bộ lặp ngược
{
    node<T> *curr;
    public:
        node<T>*&getcur(){return curr;} //Trả về con trỏ curr
        dlist_rite(node<T>*c=NULL) {curr=c;}  //Hàm tạo
        dlist_rite<T>& operator=(dlist_rite<T> const &it)  //toán tử gán
        {
            this->curr=it.curr;
            return *this;
        }
        bool operator!=(dlist_rite<T> it)  //Toán tử khác - so sánh không bằng
        {
            return curr!=it.curr;
        }
```

```
        T &operator*(){return curr->elem;}  //Toán tử lấy giá trị *it
        dlist_rite<T> operator++()  //++it
        {
            curr=curr->prev;//Gán curr là node phía trước
            return curr;//Trả về curr
        }
        dlist_rite<T> operator++(int)  //it++
        {
            node<T>*p = curr;//Tạo p bằng curr
            curr = curr->prev;//Gán curr là node phía trước
            return p;//Trả về p
        }
};
template <class T>
class dlist
{
    node<T>*Head,*Tail;//Head quản lý đầu danh sách, Tail quản lý đuôi danh sách
    int n; //số phần tử của dlist - size
    public:
        dlist() {Head = Tail = 0; n = 0;}//Hàm tạo không đối - Head=Tail=NULL
        dlist(int k,T x)//Hàm tạo có đối - k là số lượng phần tử, x là giá trị mỗi phần tử
        {
            Head = Tail = 0;//Head=Tail=NULL
            n=0;
            while(k--) push_back(x);//Them k phần tử
        }
        ~dlist()//Hàm hủy
        {
            while(Head)
            {
                node<T>*p = Head; // Tao node *p bằng Head hiện tại
                Head = Head->next; // Gán Head bằng node tiếp theo nó quản lý (Head->next)
                delete p; //Hủy node p
            }
        }
        int size() {return n;} //Trả về kích thước
        bool empty(){return n == 0;} //Kiểm tra rỗng
        T &front() {return Head->elem;} //Trả về phần tử đầU, thêm "&" để có thể vừa đọc vừa
ghi
        T &back() {return Tail->elem;}  //Trả về phần tử cuối, thêm "&" để có thể vừa đọc
vừa ghi
        T &operator[](int k)            //list trong STL không có hàm này, truy cập đến phần
tử thứ k
        {
            node<T> *p = Head;
            while(k--) p = p->next;
            return p->elem;
        }
        void push_back(T x) //Thêm vào cuối 1 phần tử
        {
            if(n==0) Head = Tail = new node<T>(x); //Nếu dlist rỗng => tạo node mới ở Tail
và gán Head=Tail
            else Tail = Tail->next = new node<T>(x,Tail,0);
            //dlist khác rỗng => tạo node mới ở đuôi có prev=Tail,next=NULL, Gán Tail bằng
node vừa mới tạo
            n++;//tăng size lên 1
        }
        void push_front(T x) //Thêm vào đầu 1 phần tử
        {
            if(n==0) Head = Tail = new node<T>(x);//Nếu dlist rỗng => tạo node mới ở Tail và
gán Head=Tail
            else Head = Head->prev = new node<T>(x,0,Head);
            //dlist khác rỗng => tạo node mới ở đầu có prev=NULL,next=Head, Gán Head bằng
node vừa mới tạo
```

```
        n++;//tăng size lên 1
    }
    void pop_back() //Xóa 1 phần tử cuối
    {
        if(n==1) {delete Head; Head = Tail = 0; n = 0; return;}
        //Nếu chỉ có 1 phần tử xóa Head,gán Head=Tail=NULL,n=0 rồi thoát
        Tail = Tail->prev; //Tail gán bằng phần tử trước
        delete Tail->next; //Xóa phần tử tiếp theo của Tail đang là phần từ cuối
        Tail->next = 0; //Gán Tail->next=NULL
        n--;            //Giảm size đi 1
    }
    void pop_front() //Xóa 1 phần tử đầu
    {
        if(n == 1) {delete Head; Head = Tail = 0; n = 0; return;}
        //nếu chỉ có 1 phần tử -> xóa Head,gán Head=Tail=NULL,n=0, rồi thoát
        Head = Head->next; //Head gán bằng phần tử sau
        delete Head->prev; //Xóa phần tử trước của Head đang là phần từ đầu
        Head->prev = 0; //Gán Head->prev=NULL
        n--;    //Giảm size đi 1
    }
    void sort(bool ok = true) //Mặc định sắp xếp tăng dần (giảm dần ok=false)
    {
        if(n<=1) return; //Nếu chỉ có 1 hoặc 0 phần tử -> ko cần sx
        for(node<T>*p = Head; p != NULL; p = p->next) //p chạy từ Head đến Tail
        for(node<T>*q = p->next; q != NULL; q = q->next) //q chạy từ p->next đến Tail
        if(q->elem < p->elem == ok) swap(p->elem, q->elem); // Đổi giá trị nếu q->elem <
p->elem (sx tăng)
    }

    typedef dlist_ite<T> iterator; //Bộ lặp xuôi
    iterator begin() {return Head;} //Địa chỉ phần tử đầu của bộ lặp xuôi là Head
    iterator end() {return NULL;}   //Địa chỉ phần tử cuối = NULL

    typedef dlist_rite<T> reverse_iterator; //Bộ lặp ngược
    reverse_iterator rbegin() {return Tail;} //Địa chỉ phần tử đầu của bộ lặp ngược là
Tail
    reverse_iterator rend() {return NULL;} //Địa chỉ phần tử cuối = NULL
    void insert(iterator it,T x) //Chèn 1 phần tử ở vị trí bất kì
    {
        node<T> *p=it.getcur();//Gán p bằng node hiện tại của iterator
        if(p==Head) return push_front(x); //Nếu p là phần tử đầu thì thêm vào đầu
        node<T> *q=p->prev; //Tạo một node mới gán bằng node trước của p
        q->next=p->prev=new node<T>(x,q,p);
        //Tạo node mới có prev là node q và next là node p
        //Rồi gán q->next và p->prev là node vừa mới tạo
        n++;//Tăng size
    }
    void erase(iterator it) //xóa 1 phần tử ở vị trí bất kì
    {
        node<T> *p = it.getcur();//Gán p bằng node hiện tại của iterator
        if(p == Head) return pop_front();//Nếu nó bằng node đầu thì pop_front
        if(p == Tail) return pop_back();//Nếu nó bằng node cuối thì pop_back
        node<T> *q = p->prev,*r = p->next;
        //Tạo q bằng node đừng trước p, tạo r là node đứng sau p
        q->next= r ; r->prev = q;
        //Gán node tiếp theo của q là r, và node phía trước của r là q
        delete p;//Xóa node p
        n--;//giảm size
    }
};

int main()
{
    dlist<int> L(5,6);
```

```
    for(int x:{7,8,9}) L.push_back(x);
    for(int x:{1,2,3}) L.push_front(x);
    cout<<"\nL: ";
    for(int i = 0; i < L.size(); i++) cout << L[i] << " ";

    L.sort(false);
    L.pop_back();
    L.pop_front();
    cout<<"\nL: ";
    for(int i = 0; i < L.size(); i++) cout << L[i] << " ";
    cout << "\nDuyet xuoi : "; for(auto z:L) cout<<z<<" ";
    cout<<"\nDuyet nguoc: ";
    for(dlist<int>::reverse_iterator it=L.rbegin();it!=L.rend();it++) cout<<*it<<" ";
    auto it = L.begin(); for(int s = 1; s <= 4; s++) it++;
    L.insert(it, 10);
    cout << "\nL: "; for(auto z:L) cout << z << " ";
    L.erase(L.begin());
    it = L.begin(); for(int s = 1; s <= 6; s++) it++;
    *it = -2;
    cout << "\nL: "; for(auto z:L) cout<<z<<" ";
    L.erase(it);
    cout<<"\nL: "; for(auto z:L) cout<<z<<" ";
}
```

Tree

Cây gia phả	Code

```
#include<bits/stdc++.h>
using namespace std;

typedef long long ll;
ll maxChild = 0, height = 0;

class node
{
public:
    vector <node*> child;
    ll num, deep;
    string name;
    node(string na  = "", ll n = 0, ll de = 0)
    {
        name = na;
        num = n, deep = de;
    }
};

// Cập nhật node theo cách duyệt trung thứ tự
void update(node *&H, string a, string b)
{
    if (H->child.size()) update(H->child[0], a, b);

    // Khi tìm thấy node a thì thực hiện gán node b là con của node a
    if (H->name == a)
    {
        node *tmp = new node(b, 0, H->deep + 1);

        H->child.push_back((node*)tmp);
        H->num++;
        return;
    }
    for (ll i = 1; i < H->child.size(); i++) update(H->child[i], a, b);
}

// Duyệt trung thứ tự để in ra cây gia phả: con cả -> gốc -> các con còn lại
void inorder2(node *&H)
{
    // đánh dấu số lượng con nhiều nhất có cùng 1 cha
    maxChild = max(maxChild, H->num);

    // lưu độ sâu lớn nhất trong cây
    height = max(H->deep, height);

    //Nếu node H có con thì thực hiện duyệt con cả trước
    if (H->child.size()) inorder2(H->child[0]);
    cout << H->name << " ";

    // Sau đó duyệt nốt các con còn lại
```

```
    for (ll i = 1; i < H->child.size(); i++) inorder2(H->child[i]);
}

int main()
{
    ll n; cin >> n;

    // y là con là x
    string x, y; cin >> x >> y;

    // Tạo nút gốc, tên: x, số con: 0, độ sâu: 1
    node *root = new node(x, 0, 1);

    // Đứa con đầu của gốc, tên: y, số con: 0, độ sâu: 2
    node *tmp = new node(y, 0, 2);

    // Thêm vào danh sách con của gốc và tăng số con của gốc thêm 1
    root->child.push_back(tmp);
    root->num++;

    // Trừ 2 vì đã có thao tác tạo nút gốc cùng con của nó, và n người thì chỉ có n-1 quan
hệ
    n -= 2;
    while(n--)
    {
        // Nhập vào tên của 2 người a, b
        string a, b; cin >> a >> b;
        // Cập nhật thêm b là con của a
        update(root, a, b);
    }

    inorder2(root);
    cout << endl;
    cout << maxChild << endl << height;
}
```

Xây dựng danh bạ	Code

```
#include<bits/stdc++.h>
using namespace std;
/*ý tưởng tạo 1 cây tiền tố có elem là số chuỗi đi qua và 1 mảng 26 phần tử
    tương ứng mới 26 chữ cái
    - mỗi khi thêm một chữ cái ta sẽ đi từ gốc qua các node tương ứng với các chữ (vd root :
r->o->o->t)
    nếu chưa có ta tạo mới nếu có rồi ta tăng elem
    - nếu tìm kiếm ta tìm theo từ vừa nhập theo cây nếu đi hết thì trả về elem ngược trả về
0*/
struct node
{
    int elem; //Số chuỗi đi qua
    node*child[26] = {};
    // Tạo mảng child có kiểu *node gồm 26 phần tử tương ứng 26 chứ cái a->z
    node(){elem = 1;}
};
void update(node *&H,char *p)//Hàm cập nhật - đầu vào là node và con trỏ chứa địa chỉ kí tự
đó trong chuỗi
{
    if(!H) H = new node();//Nếu node = NULL thì tạo node mới
    else H->elem++;//Nếu đã có tăng thêm 1
    if(*p) update(H->child[*p - 'a'], p+1);
    //Tiếp tục gọi đệ quy hàm update cho đến khi hết p=NULL
    //H->child[*p-a] : truyền node con ứng với kí tự *p có chỉ số *p-'a'
    //(Ví dụ:*p='b' =>'b'-'a' =1 => H->child[1])
    //p+1: truyền địa chỉ kí tự tiếp theo
}
int get(node*H, char *p)//Hàm tìm xem ứng với chuỗi truyền vào có bao nhiêu chuỗi t/m
{
    if(!H) return 0;//Nếu H=NULL => không có chuỗi nào thỏa mãn => trả về 0
    if(*p == 0) return H->elem; //Nếu *p==0 => đã chạy hết chuỗi => trả về số chuỗi đã đi
qua
    return get(H->child[*p - 'a'], p+1);
    //Nếu *p chưa chạy hết => gọi đệ quy với node(H->child[*p-'a']) và kí tự tiếp theo (p+1)
}
int main()
{
    node *root = nullptr; //new node();
    int n;//Số truy vấn
    char q[10], x[1000];//mảng char
    scanf("%d", &n);
    while(n--)
    {
        scanf("\n%s %s", q, x);
        //\n%s : nhập q (kiểu thao tác: add - find) cho đến khi gặp kí tự cách
        //%s : nhập chuỗi x
        if(q[0] == 'a') update(root, x);
```

```
        //Nếu p[0]=='a' => p==a"add" => gọi update: truyền root và địa chỉ phần tử đầu (x)
        else printf("%d\n", get(root, x));
        //else =>p=="find" => gọi hàm get :truyền root và địa chỉ phần tử đầu (x)
    }
}
```

BinaryTree

Cây Heap và Ứng dụng trong cài Priority Queue	Code

```
#include<bits/stdc++.h>
using namespace std;
template <class T>
struct node
{
    T elem;
    int n;
    //n size nó bằng 1 + size(left)+size(right) - số lượng node nhỏ hơn nó mà nó quản lý
    node *left,*right;
    node(T x,node<T>*L=0,node<T>*R=0,int _n=1)//mặc định left=right =NULL và n=1
    {
        n=_n;
        elem=x;
        left=L;
        right=R;
    }
};
template<class T,class CMP>//CMP là hàm truyền vào
void update(node<T> *&H,T x,CMP ss)//Hàm cập nhật phần tử, ss là tên hàm
{
    if(!H) H=new node<T>(x);
    //Nếu node = NULL => Chưa có  => Tạo node mới
    else if(ss(H->elem,x))H=new node<T>(x,H,0,H->n+1);
    //Nếu hàm ss trả về đúng thì tạo một node mới
    //Node mới có elem =x, left=H, right = NULL, n= H->n (size(left)) + 0 (size(right)) + 1
    //Gán H = node vừa mới tạo
    else
    {
        H->n++;//size tăng 1
        if(!H->left) H->left=new node<T>(x);//Nếu left chưa có gì tạo mới rồi gắn bằng left
        else if(!H->right) H->right=new node<T>(x);//Nếu right chưa có gì tạo mới rồi gắn
bằng right
        else update(H->left->n<H->right->n?H->left:H->right,x,ss);
        //So sánh xem bên náo ít node hơn thì gọi update về phía bên đó
        //-> giúp cây được cân bằng hơn về 2 bên
        //Nếu = thì gọi về bên phải
    }
}
template<class T,class CMP>//CMP là hàm truyền vào
void remove(node<T> *&H,CMP ss)//Hàm xóa phần tử đỉnh của heap, ss là tên hàm
{
    if(!H) return; //Nếu NULL thì thoát
    if(!H->left) H=H->right; //Nếu cây bên trái rỗng H gán = cây bên phải
    else if(!H->right) H=H->left; //Nếu cây bên phải rỗng H gán = cây bên trái
    else//Nếu cả hai bên không rỗng
    {
        H->n--;//Giảm size đi 1
        if(ss(H->left->elem,H->right->elem)) {H->elem=H->right->elem;remove(H->right,ss);}
        //Nếu hàm ss đúng H->elem bằng elem của cây bên phải (H->right->elem)
        //và gọi đệ quy xóa phần từ cây bên phải
        else {H->elem=H->left->elem;remove(H->left,ss);}
        //Nếu hàm ss sai H->elem bằng elem của cây bên trái (H->left->elem)
        //và gọi đệ quy xóa phần từ cây bên trái
    }
}
template <class T,class CMP=less<T> >
class Priority_Queue  //Hàng đợi ưu tiên
{
    node<T> *root=0;//node đỉnh- node gốc
    CMP ss;//Hàm xác đinh ưu tiên (lơn hơn, nhỏ hơn, ...)
    public:
        Priority_Queue() {root=0;}//Hàm tạo - root = NULL
        int size(){if(!root) return 0; return root->n;}
        //Trả về root->n do nó là phần tử quản lý tất cả phần tử còn lại
        bool empty(){return root==NULL;}//Kiểm tra  có phần tử không
        void push(T x) {update(root,x,ss);}//Thêm vào hàng đợi
        void pop() {remove(root,ss);}//Xóa khỏi hàng đợi phần tử ưu tiên nhất
        T top() {return root->elem;} //Trả về phần tử trên đỉnh - chỉ có thể đọc
};
int main()
{
    Priority_Queue<string,greater<string> > Q;
    for(auto x:{"chi pheo","lao hac","thi no","thang muc","ba kien","thang xien","cau
vang"}) Q.push(x);
    while(Q.size())
    {
```

```
        cout<<Q.top()<<"\n";
        Q.pop();
    }
}
```

Set

Cài đặt Set bằng cây tìm kiếm nhị phân	Code

```
#include<bits/stdc++.h>
using namespace std;
template <class T>
struct node
{
    T elem;
    node*left,*right,*next;//tạo 2 node quản lý trái, phải,next là node tiếp theo
    node(T e) {elem=e;left=right=next=0;}
};
template <class T,class CMP>
int update(node<T>*&H,int x,CMP ss)    //tra ve 1 neu them duoc, tra ve 0 neu da co va khong
them
{
    if(!H) {H=new node<T>(x); return 1;}//nếu chưa có thì tạo mới trả về 1 là 1 phần tử được
thêm vào
    if(H->elem==x) return 0;//nếu có rồi thì trả về 0 là không cần thêm
    return update(ss(x,H->elem)?H->left:H->right,x,ss);
    //nếu chưa tìm thấy gọi đệ quy về trái ỏ phải tùy theo hàm ss
}
template <class T>
void inorder(node<T> *H,list<node<T> *> &L)//duyệt ttt tạo bộ lặp - list phải có &
{
    if(H)//nếu H có giá trị
    {
        inorder(H->left,L);//gọi node trái
        L.push_back(H);//Thêm vào list
        if(L.size()==2) {L.front()->next=L.back(); L.pop_front();} //mang xong
        //nếu list có 2 phần từ liên kết chúng lại : L.front()->next=L.back()
        //sau đó bỏ phần từ đầu
        inorder(H->right,L);//Gọi bên phải
    }
}
template <class T>
int Max(node<T>*H)//Hàm tìm max
{
    if(!H) return -INT_MAX; //nếu null thì trả về âm vô cùng
    return H->right?Max(H->right):H->elem;
    //nếu H->right mà tồn tại -> gọi về đệ quy bên phải trả về lớn hơn
}
template <class T>
int Min(node<T>*H)//Hàm tìm min
{
    if(!H) return INT_MAX; //nếu null thì trả về dương vô cùng
    return H->left?Min(H->left):H->elem;
    //nếu H->left mà tồn tại -> gọi về đệ quy bên phải trả về nhỏ hơn
}
template <class T,class CMP>
int remove(node<T> *&H,int x,CMP ss)//Hàm xóa
{
    if(!H) return 0;//nếu null -> ko có trả về 0 là 0 xóa phần từ nào
    if(H->elem!=x) return remove(ss(x,H->elem)?H->left:H->right,x,ss);
    //chưa tìm thấy gọi đệ quy trái hoặc phải
    if(!H->left) H=H->right;//nếu bên trái rỗng H = vế phải
    else if(!H->right) H=H->left;//bên phải rỗng  H= vế trái
    else //nếu cả 2 bên có cây
    {   //ý tưởng chuyển node max bên trái or min bên phải lên thay thế
        H->elem=Max(H->left);//Gán H->elem là max của cây bên phải
        remove(H->left,H->elem,ss);//Xóa node có elem=H->elem ở bên phải
    }
    return 1;
}
template <class T>
class node_ite  //Bo lap xuoi
{
    node<T> *curr;
    public:
        node<T>*&getcur(){return curr;}
        node_ite(node<T>*c=NULL) {curr=c;}  //hàm tạo
        node_ite<T>& operator=(node_ite<T> const &it)  //toán tử gán
        {
            this->curr=it.curr;
            return *this;
        }
        bool operator!=(node_ite<T> it)  //só sánh không bằng
        {
            return curr!=it.curr;
        }
```

```
        T &operator*(){return curr->elem;}  //toán tử lấy giá trị *it
        node_ite<T> operator++()  //++it
        {
            curr=curr->next;
            return curr;
        }
        node_ite<T> operator++(int)  //it++
        {
            node<T>*p=curr;
            curr=curr->next;
            return p;
        }
};
template <class T,class CMP=less<T> >
class SET
{
    node<T>*root=0;   //goc cay BST
    int n=0;          //so phan tu trong cay
    CMP ss;
    public:
        SET() {n=0; root=0;}
        int size(){return n;}
        bool empty() {return n==0;}
        void insert(T x) {n+=update(root,x,ss);} //nếu thêm phần tử n+=1 nếu có rồi n+=0
        void erase(T x) {n-=remove(root,x,ss);} //nếu xóa được phần tử n-=1 nếu không có
n-=0
        typedef node_ite<T> iterator;//bộ lawpk xuôi
        iterator begin()
        {
            list<node<T>*> L;
            inorder(root,L);//tạo lại liên kết các node theo ttt
            node<T>*p=root; //
            while(p->left) p=p->left; // tìm phần tử trái nhất là begin
            return p;
        }
        iterator end() {return NULL;}
};
int main()
{
    SET<int > S;
    for(int x:{4,7,2,8,4,8,3,2}) S.insert(x);
    cout<<"\n S: ";for(auto s:S) cout<<s<<" ";
    S.erase(4);
    cout<<"\n S: ";for(auto s:S) cout<<s<<" ";
    cout<<"\nsize : "<<S.size();
}
```

Priority Queue

Cài Hàng đợi ưu tiên bằng mảng	Code

```
#include<bits/stdc++.h>
using namespace std;
template<class T, class CMP = less<T> >
class PQ
{
    T *a;
    int cap, n;  //capacity, size
    CMP ss;
    private:
        void heapy(int n, int k)  //vun tại vị trí k trong dãy a[0]..a[n-1]
        {
            if(2*k+1>=n) return;
            int p=2*k+1;            //xét con 1
            if(p+1<n && ss(a[p],a[p+1])) p++; //nếu con 2 lớn hơn chuyển p sang con 2
            if(ss(a[k],a[p])) {swap(a[k],a[p]);heapy(n,p);}
        }

    public:
        PQ() {a = 0; cap = n = 0;}
        ~PQ() {if(a) delete a;}
        int size(){return n;}
        bool empty() {return n == 0;}
        T top() {return a[0];}//lấy phần tử đỉnh - không thể thay đổi giá trị
        void push(T x)//thêm phần tử
        {
            if(n == cap)//sửa chứa đầy hoặc chưa có
            {
                cap = cap*2+1;//tạo lại cap
                T*tem = a;//sao chép mảng
                a=new T[cap];//cấp phát lại
                for(int i = 0; i < n; i++) a[i] = tem[i];//gán lại vào a
                if(!tem) delete []tem;//huy
            }

            a[n++] = x;//thêm phần tử rồi tăng size
```

```
            int k = n-1;//
            while(k > 0 && ss(a[(k-1)/2], a[k]))
            {
                swap(a[(k-1)/2], a[k]);
                k = (k-1)/2;
            }

            //cập nhập lại giá trị
            //làm ngược lại với vun đống
            //a[(k-1)/2] là cha a[k] là con
        }
        void pop(){a[0] = a[--n];   heapy(n,0);}//giảm size - vun lại đống
};
int main()
{
    PQ <int,greater<int> > Q;
    for(auto x:{52, 86, 54, 6, 68, 8, 26, 98, 987, 408, 62, 56, 36, 26})
        Q.push(x);
    while(Q.size())
    {
        cout << Q.top() << " ";
        Q.pop();
    }
}
```

Hash Table

Cài bảng băm	Code

```
#include<bits/stdc++.h>
using namespace std;
template<class T>
class HT // ứng dụng trong unordered_set va unordered_map
{
    int M = 13;  //số ngăn của mảng
    int n; //size
    list<T> *L; //tạo M list để lưu phần tử
    int hashfunction(T x) //Hàm băm
    {
        hash<T> H;//dùng có sẵn trong std
        return H(x)%M;//trả về chia dư cho số ngăn -> ra ngăn chứa nó
    }
    public:
        HT(int _M = 13) { n =  0; M = _M; L = new list<T>[M];}//Hàm tạo
        int size(){return n;}
        bool empty() {return n == 0;}
        void insert(T x)//Thêm phần tử vào mảng
        {
            int k = hashfunction(x);//k là số sau khi qua hàm băm -> chỉ số của list
            L[k].push_back(x);//thêm vào cuối
            n++;//tăng size
        }

        bool find(T x)
        {
            int k = hashfunction(x);//k là số sau khi qua hàm băm -> chỉ số của list
            for(auto it = L[k].begin(); it != L[k].end(); it++)
                if(*it == x) return true;
            //chạy hết list tìm phần tử nếu tìm thấy trả về true, ngược lại trả về false
            return false;
        }

        void erase(T x)
        {
            int k = hashfunction(x);//k là số sau khi qua hàm băm -> chỉ số của list
            auto it = L[k].begin();//it là phần tử đầu của list / auto=list<T>::iterator
            while(it != L[k].end() && *it != x) it++;//tìm đến khi nào thấy hoặc hết danh
sách
            if(it == L[k].end()) return;//chạy hết -> ko thấy -> thoát
            L[k].erase(it);//tìm thấy -> xóa
            n--; //giảm size
        }

        void travel()//hàm in các phần tử của list
        {
            for(int i = 0; i < M; i++)//chạy từng list
            for(auto x : L[i]) cout << x << " ";//in từng phần tử của list đó
        }
};

int main()
{
    HT<int> H;
    for(int x : {41, 13, 52, 5, 15, 23, 6, 35, 65, 36, 3, 65, 26, 3, 6, 51, 25, 234, 14, 67,
9})
        H.insert(x);
```

```
    cout << "H : "; H.travel();
}
```

Giải thuật
BFS

Tìm đường đi ngắn nhất trong mê cung	Code

```
#include<bits/stdc++.h>
using namespace std;
class mecung
{
    int n, m, A[105][105];

    //Vị trí điểm xuất phát và kết thúc
    int sx, sy, fx, fy;
    public:
        void sol()
        {
            // Nhập input
            cin >> n >> m;
            for(int i = 1; i <= n; i++)
                for(int j = 1; j <= m; j++)
                    cin >> A[i][j];
            cin >> sx >> sy >> fx >> fy;

    // Tạo 1 hàng rào (toàn số 1) xung quanh mê cung
    for(int i = 0; i <= n+1; i++)
        A[i][0] = A[i][m + 1] = 1;
    for(int j = 0; j <= m+1; j++)
        A[0][j] = A[n + 1][j] = 1;

    cout << BFS(sx, sy, fx, fy);
}

int BFS(int sx, int sy, int fx, int fy)
{
    queue <pair<int, int>> Q; //Chứa tọa độ các điểm vừa đi qua

    // Đưa điểm xuất phát vào queue
    Q.push({sx, sy});

    // Đánh dấu ô này không đi được nữa, đồng thời cũng để đếm số bước đi
    A[sx][sy] = 1;

    while(Q.size())
    {
        //x, y: tọa độ của ô đang xét
        int x = Q.front().first;
        int y = Q.front().second;
        Q.pop();

        // Danh sách 4 ô xung quanh ô đang xét
        pair <int, int> Next[] = {{x, y + 1}, {x,y - 1},
                                  {x - 1, y}, {x + 1, y}};

        // Lần lượt xét 4 ô xung quanh
        for(auto v : Next)
            // Nếu ô này đi được
            if(A[v.first][v.second] == 0)
            {
                // Số bước đi đến ô này = số bước đi đến ô trước đó + 1
                A[v.first][v.second] = A[x][y] + 1;
                Q.push(v); // Thêm vào queue để sau này duyệt tiếp
            }

        // Nếu ô đích != 0 (có cách để đi đến) thì in ra số bước đi đến ô này
        // Trừ đi 1 vì điểm xuất phát ta đánh dấu là 1 (bước)
        // Không đánh dấu điểm xuất phát là 0 được, vì nó sẽ trùng với những ô
có thể đi được
        if(A[fx][fy] != 0) return A[fx][fy] - 1;
    }
    return -1;
}
};
int main()
{   mecung M;
    M.sol();
}
```

Nhóm bạn	Code

```cpp
#include<bits/stdc++.h>
using namespace std;
class Group
{
    int n, m, d[100005] = {}; //d[i]: đánh dấu xem người i đã được duyệt chưa
    int k = 0, res = 0; // k: số khu vực, res: số người của khu vực đông nhất
    vector <int> A[100005]; //A[i]: danh sách những người quen với i
    public:
        void sol()
        {
            cin >> n >> m;
            for(int i = 1; i <= m; i++)
            {
                int u, v;
                cin >> u >> v;

                // u quen v và v cũng quen u
                A[u].push_back(v);
                A[v].push_back(u);
            }

            for(int i = 1; i <= n; i++)
                if(d[i] == 0)
                {
                    k++;
                    int z = BFS(i);
                    if(res < z) res = z;
                }
        cout << k << "\n" << res;
}

//Tìm tất cả những người quen của s và trả về số người của nhóm đó
int BFS(int s)
{
    int dem = 1;
    queue <int> S;

    // đánh dấu người s đã duyệt rồi
    d[s] = 1;
    S.push(s);

    while(S.size())
    {
        int u = S.front(); S.pop();

        // Duyệt danh sách những người quen của u
        for(int v : A[u])

        // Nếu người v chưa được duyệt
        if(d[v] == 0)
        {
            // Đánh dấu người v đã duyệt rồi
            d[v] = 1;
            dem++;

            // Đưa vào hàng đợi để sau này duyệt tiếp
            S.push(v);
                }
            }
            return dem;
        }
};
int main()
{
    Group G;
    G.sol();
}
```

Đong nước	Code

```cpp
/*Cho hai bình có sức chứa là n và m lít hãy tìm cách đong k lít nước
biết rằng mỗi lần đong có thể đổ hết nước trong một bình đi,
hoặc đổ thêm nước vào một bình, hoặc đổ nước từ bình nọ sang bình kia và
cho số lượng nước để sử dụng là vô hạn và ban đầu hai bình đều chưa chứa nước.*/

/*Ý tưởng: tạo queue S lưu lượng nước của 2 bình, map lưu số bước hiện tại D.
Khi đong lượng nước k sẽ có 6 trường hợp xảy ra như code dưới, xét từng trường hợp
nếu thỏa mãn sẽ thêm vào S và tăng D+1.
Chạy đến khi lượng nước 1 trong 2 bình được k sẽ dừng.*/
#include<bits/stdc++.h>
```

```cpp
using namespace std;
typedef pair<int, int> TT;
class water
{
    int n, m, k;
    int BFS() //tìm theo bề rộng xét các node theo thứ tự ưu tiên từ nhỏ đến lớn
    {
        queue <TT> S; // Lưu lượng nước của bình 1 và bình 2
        map <TT, int> D;// số bước thực hiện
        S.push({0, 0});    //{0,0} make_pair(0,0) TT(0,0)
        D[{0,0}] = 0;
        while (S.size())
        {
            int x = S.front().first, y = S.front().second, z = x + y;
            S.pop();
            //6 trường hợp đong nước vào 2 bình có thể xảy ra:
            TT Next[6] = {{0, y}, {x, 0},
                          {n, y}, {x, m},
                          {max(0, z - m), min(z, m)},
                          {min(z, n), max(0, z - n)}};
            for(auto v : Next)
                //nếu trường hợp nào khả thi sẽ cất vào S và tăng D+1
                if(D.find(v) == D.end())
                {
                    S.push(v);
                    D[v] = D[{x, y}] + 1;
                    //nếu 1 trong 2 bình bằng k sẽ return D hiện tại
                    if(v.first == k || v.second == k) return D[v];
                }
        }
        return -1;//nếu tất cả các trường hợp không thỏa mãn return -1
    }

    public:
        void sol()
        {
            cin >> n >> m >> k;
            int res = BFS();
            if(res == -1) cout << "Khong dong duoc nuoc";
            else cout << res;
        }
};
int main()
{
    water W;
    W.sol();
}
```

Hiển thị các bước đong nước	Code

```cpp
#include<bits/stdc++.h>
using namespace std;
typedef pair<int, int> TT;
class water
{
    int n, m, k;
    map<TT, TT> D;  //tạo mảng cha D[{3,2}]={0,5}
    void INKQ(TT f)
    {
        if(f.first == 0 && f.second == 0)
            cout << "(0, 0)";
        else
        {
            INKQ(D[f]);
            cout << "->(" << f.first << "," << f.second << ")";
        }
    }

    void inkq(TT f)
    {
        stack <TT> S;
        do
        {
            S.push(f);
            f = D[f];
        }
        while(f.first || f.second);
        cout << "(0,0)";
        while(S.size())
        {
            cout << "->(" << S.top().first << "," << S.top().second << ")";
            S.pop();
        }
    }

    void DFS()
    {
```

```
        stack<TT> S;
        S.push({0,0});    //{0,0} make_pair(0,0) TT(0,0)
        D[{0, 0}] = {-1, -1}; //coi cha của (0,0) la (-1,-1)
        while (S.size())
        {
            int x = S.top().first, y = S.top().second, z = x + y;
            S.pop();
            TT Next[6] = {{0, y}, {x, 0},
                          {n, y}, {x, m},
                          {max(0, z - m), min(z, m)},
                          {min(z, n), max(0, z - n)}};

            for(auto v : Next)
            if(D.find(v) == D.end())
            {
                S.push(v);
                D[v] = {x, y};
                if(v.first == k|| v.second == k)
                {
                    // INKQ(v);
                    inkq(v);
                    return;
                }
            }
        }
        cout<<"\nKhong dong duoc nuoc";
    }

    public:
        void sol()
        {
            cin >> n >> m >> k;
            DFS();
        }
};

int main()
{
    water W;
    W.sol();
}
```

Phân loại các F để cách ly Covid-19	Code

```
/*Ý tưởng:tạo 1 mảng các vector trong đó vector chứa những người đã tiếp xúc với đối tượng
mảng chứa
tạo 2 mảng d,F chứa số k vd:d[x]=2 thì x bị F2
tạo queue Q chứa số lượng người bị covid và các F
nếu ai có tiếp xúc với F thì push vào Q để xét tiếp và tăng số F đang xét +1*/
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n, m, x, u, v;
    cin >> n >> m;
    vector<int> A[n+5];  //Mang cac vector
    while(m--)
    {
        cin >> u >> v;
        A[u].push_back(v);
        A[v].push_back(u);
    }
    int d[n + 5]; fill(d,d+n+1,-1);  //d[x]=k thi x la F[k]
    int F[n + 5] = {};                 //so F[i] ban dau toan 0
    queue<int> Q;
    cin >> F[0];
    for(int i = 1; i <= F[0]; i++)
    {
        cin>>x;
        Q.push(x);
        d[x] = 0;
    }

    while(Q.size())
    {
        int u = Q.front(); Q.pop();
        for(int v : A[u])
            if(d[v] == -1)
            {
                d[v] = d[u]+1;
                Q.push(v);
                F[d[v]]++;
            }
    }
    for(int i =0 ; F[i] != 0; i++)
```

```
        cout << "F" << i << ": " << F[i] << "\n";
}
```

Sort

Bubble Sort	Code

```
#include<bits/stdc++.h>
using namespace std;
void bubble(int *a, int n)  //sx a[0]...a[n-1]
{ // sx tăng dần
    for(int i = 0; i < n; i++)//Duyệt i : 0->n-1
    {
        int ok = 1;//ok : kiểm tra xem có phải swap làn nào không nếu không thì thoát
        for(int j = n-1; j > i; j--)//Duyệt j: (n-1) -> (i+1)
        if(a[j-1] > a[j]) {swap(a[j-1], a[j]); ok = 0;}//nếu a[j-1]>a[j] thì đổi chỗ và ok=0
        if(ok) return;// Nếu ok =1-> không đổi chỗ -> thoát ,ok=0 thì tiếp tục
    }
}
void bubble2(int *L, int *R)// sx từ vị trí L đến R-1 -  L,R đầu vào là con trỏ
{
    for(int *p = L; p < R; p++) // duyệt con trỏ p: L->R-1
    {
        int ok = 1;//ok : kiểm tra xem có phải swap làn nào không nếu không thì thoát
        for(int *q = R-1; q > p; q--)//duyệt *q : R-1 -> L+1
        if(*q < *(q-1)) {swap(q[-1], q[0]); ok = 0;}
        //phần tử trước (p-1) > phần tử sau (p) -> swap hai giá trị này , ok =0
        if(ok) return;// Nếu ok =1-> không đổi chỗ -> thoát ,ok=0 thì tiếp tục
    }
}
template <class T,class CMP = less<T>>//CMP là hàm so sánh
void bubble3(T *L,T *R,CMP ss = less<T>() )// sx từ vị trí L đến R-1 - L,R đầu vào là con
trỏ
{
    for(T *p = L; p < R; p++)// duyệt con trỏ p: L->R-1
    {
        int ok = 1;//ok : kiểm tra xem có phải swap làn nào không nếu không thì thoát
        for(T *q = R-1; q > p; q--)//duyệt *q : R-1 -> L+1
        if(ss(*q, *(q-1))) {swap(q[-1], q[0]); ok = 0;}
        //Nếu hàm so sánh trả về đúng -> swap hai giá trị này , ok =0
        if(ok) return;// Nếu ok =1-> không đổi chỗ -> thoát ,ok=0 thì tiếp tục
    }
}
int main()
{
    int a[] = {3, 12, 0, -3, 7, -12, 102, -9, 32, 8}, n =
sizeof(a)/sizeof(int);//a=sizeof(int) * số phần tử
    //bubble(a + 4, 3);
    //bubble2(a, a + n);
    bubble3(a, a+n, greater<int> ());//greater -> sx giảm dần
    for(auto x:a) cout << x << " ";
}
```

Insertion Sort	Code

```
#include<bits/stdc++.h>
using namespace std;
void INSERT(int *a,int n)  //sx a[0]...a[n-1]
{
    for(int i=1;i<n;i++)//duyệt i: 1 -> n-1
    {
        int j,x=a[i]; //Tạo j,x gán bằng giá trị vị trí đang xét
        for(j=i-1;j>=0 && x<a[j];j--) //duyệt từ i-1 -> 0 hoặc
        a[j+1]=a[j];
        // Chèn số vào vị trí thích hợp
        // sx giảm  : chừng nào a[j]>x -> dịch về cuối 1 bước-> a[j+1]=a[j]
        // Nếu a[j]<=x -> tìm được vị trí chèn -> thoát
        a[j+1]=x;//Chèn a[j+1]=x
    }
}
void Insert(int *L,int *R)// sx từ vị trí con trỏ L -> R-1
{
    for(int *p=L+1;p<R;p++)// p : duyệt từ L+1 ->R-1
    {
        int *q,x=*p; //Tạo *q , gán x = giá trị của *p
        for(q=p-1;q>=L && x<*q;q--) //duyệt q :p-1 -> L hoặc  đến x<*q thì dừng
        q[1]=q[0];//*(q+1)=*q; - dịch về cuối 1 bước
        // sx giảm  : chừng nào *q>x -> dịch về cuối 1 bước -> q[1]=q[0]
        // Nếu *q<=x -> tìm được vị trí chèn -> thoát
        *(q+1)=x;//chèn *(p+1) = x
    }
}
template <class T,class CMP=less<T>>
void insert(T *L,T *R,CMP ss=less<T>() )// sx từ vị trí con trỏ L -> R-1
{
```

```
    for(int *p=L+1;p<R;p++) //duyệt q :p-1 -> L
    {
        T *q,x=*p; //Tạo *q , gán x = giá trị của *p
        for(q=p-1;q>=L && ss(x,*q);q--)
        //duyệt q :p-1 -> L đến chừng nào ss trả về true thì dừng
        //Nếu *q<=x -> tìm được vị trí chèn -> thoát
        q[1]=q[0]; //Nếu không  dịch về cuối 1 bước
        *(q+1)=x;//chèn *(p+1) = x
    }
}
int main()
{
    int a[]={253,166,754,768,525,854,625,845,363},n=sizeof(a)/sizeof(int);
//  insert(a,a+n);
    insert(a,a+n,greater<int>());
    for(auto x:a) cout<<x<<" ";
}
```

SelectionSort	Code

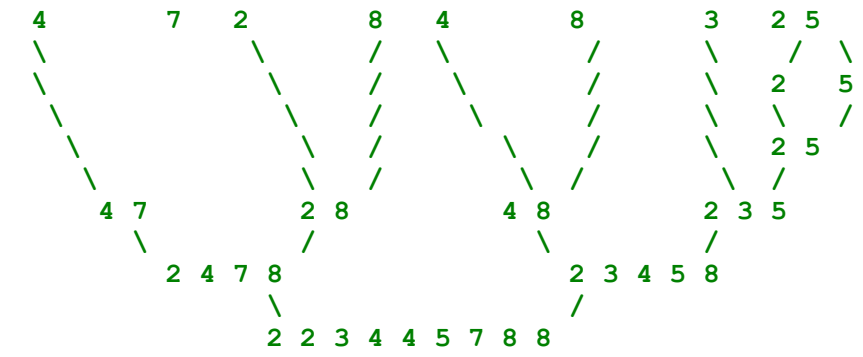
```
#include<bits/stdc++.h>
using namespace std;
void SELECT(int *a,int n)  //sx a[0]...a[n-1]
{   //sx tăng dần
    for(int i=0;i<n;i++)// i : 0-> n-1
    {
        int p=i;//Gán p= i là vị trí đang xét - p lưu vị trí phần tử min
        for(int j=i+1;j<n;j++) //j:i+1->n-1
        if(a[j]<a[p]) p=j;//Tìm min từ i+1 -> n-1
        swap(a[i],a[p]); //Swap giá trị a[i] và a[p];
    }
}
void Select(int *L,int *R)// sx từ vị trí L -> R-1
{
    for(int *p=L;p<R;p++) // duyệt con trỏ p: L->R-1
    {
        int *z=p; //Gán z=p - z là con trỏ của min
        for(int *q=p+1;q<R;q++)  //duyệt q: p+1->R-1
        if(*q<*z) z=q; //tìm min trong đoạn và gán z=min
        swap(*p,*z);//swap p và z
    }
}
template <class T,class CMP=less<T>>
void select(T *L,T *R,CMP ss=less<T>() )// sx từ vị trí L -> R-1
{
    for(T *p=L;p<R;p++) // duyệt con trỏ p: L->R-1
    {
        T *z=p;////Gán z=p - z là con trỏ của min
        for(T *q=p+1;q<R;q++) //duyệt q: p+1->R-1
        if(ss(*q,*z)) z=q; //tìm min/max trong đoạn và gán z=min
        swap(*p,*z);//swap p và z
    }
}
int main()
{
    int a[]={253,166,754,768,525,854,625,845,363},n=sizeof(a)/sizeof(int);
//  select(a,a+n,greater<int> ());
    select(a,a+n);
    for(auto x:a) cout<<x<<" ";

}
```

MergeSort	Code

```
#include<bits/stdc++.h>
using namespace std;
/*
    # ý tưởng merger sort
    chia để trị
    - th1: suy biến (0 có phần tử) L>=R-1 thì chỉ có <=1 => xong
    - th2: L<R-1
       1. chia: chia mảng thành 2 nửa M=L+(R-L)/2
       2. tri: sắp xếp tăng dần a[L] ... a[M-1]
               sắp xếp tăng dần a[M] ... a[R-1]
       3. liên kết: trộn 2 nửa bằng nhau

       Minh họa
                                0 1 2 3 4 5 6 7 8
                                4 7 2 8 4 8 3 2 5
                               /                 \
                          4 7 2 8                 4 8 3 2 5
                         /       \               /         \
                      4 7         2 8           4 8         3 2 5
                     /   \       /   \         /   \       /   \
```




```
*/
int b[135525];
//Mảng b tạo ra để trộn trong mersort do không nên tạo mảng trong hàm đệ quy gây tốn bộ nhớ
void Mergesort(int *a,int L,int R) //sx a[L]..a[R-1]
{
    if(L>=R-1) return;  //suy biến thì thoát (<1)
    int M=(L+R)/2; // M bằng phần tử giữa của L và R // M =L+(R-L)/2;
    Mergesort(a,L,M);//Gọi nửa trái từ L -> M-1
    Mergesort(a,M,R);//Gọi nửa phía từ M -> R-1
    //int b[R+5];

    //Trộn hai nửa đã sx với nhau vào mảng b
    for(int i=L,j=M,k=L;k<R;k++)
    // duyệt k: L -> R ,k duyệt cho b, duyệt i cho bên trái, j cho bên phải
    b[k]= j>=R || (i<M && a[i]<a[j])?a[i++]:a[j++];
    //Nếu j>=R(chạy hết nửa phải) || (i<M && a[i]<a[j]) -> b[k]=a[i] -> i++
    //Ngược lại b[k]=a[j] -> j++

    for(int k=L;k<R;k++) a[k]=b[k];
    //Gán lại giá trị mảng b đã sx vào a từ L->R-1
}
template <class T,class CMP>
void mergesort(T *L,T *R,T *b,CMP ss) //sx con trỏ L đến con trỏ R-1,
{
    if(L>=R-1) return;  //suy biến thì thoát (<1)
    T *M=L+(R-L)/2; // Là đoạn giữa của L và R
    mergesort(L,M,b,ss);//Gọi nửa trái từ L -> M-1
    mergesort(M,R,b,ss);//Gọi nửa phía từ M -> R-1
    for(T *i=L,*j=M,*k=b;k<b+(R-L);k++)
    // duyệt con trỏ k : b -> b+(R-L) , i duyệt bên trái từ L, j duyệt bên phải từ M
        *k=j>=R || (i<M && ss(*i,*j))?*i++:*j++;
        //Nếu j>=R(chạy hết nửa phải) || (i<M && ss(i,j) đúng) -> *k=*i gán giá trị -> i++
        //Ngược lại *k=*j gán giá trị -> j++
    for(int *k=L,*p=b;k<R;k++) *k=*p++;
    //Gán lại giá trị mảng b đã sx vào k từ L->R-1
}
template <class T,class CMP =less<T>>
void ms(T *L,T *R,CMP ss=less<T>())
{
    T *b=new T [R-L+5];//Tạo mảng b tránh tràn bộ nhớ
    mergesort(L,R,b,ss);//Gọi mersort với hàm ss
    delete []b;//Xóa mảng b
}
int main()
{
    int a[]={23,64,74,68,38,78,86,62,43,28,39,18},n=sizeof(a)/sizeof(int);
    ms(a,a+n,greater<int>());
    for(auto z:a) cout<<z<<" ";
}
```

QuickSort	Code

```
#include<bits/stdc++.h>
using namespace std;
/*  # ý tưởng quick sort
    chia để trị
    * Do C.A.Hoarce đề xuất 1980
    - th1: suy biến (0 co phần tử) L>=R-1 thì chỉ có <=1 => xong
    - th2: L<R-1
        1. Chia
            + Chọn x bất kì thuộc a[L] ... a[R-1] làm phần tử chốt
            + Phân hoạch a[L] .. a[R] sao cho
            L    M     R
            <x | x | >=x
        2.Trị:  Đệ quy sx nửa trái nhỏ hơn x a[L] ... a[M-1]
                Đệ quy sx nửa phải lớn hơn x a[M] ... a[R]
        3.Liên kết: không làm gì

    # ý tưởng code:
    - Chọn phần tử giữa là chốt
    - Đổi chốt với phần tử đầu -> phần tử L là chốt
    - Tạo biến i => đoạn L+1 -> i là nhỏ hơn chốt, i+1 -> R-1 là lớn hơn chốt
    - chốt | <chốt | >=chốt
```

```
    - duyệt từ L+1->R-1 nếu gặp ptu nhỏ hơn(a) tăng i lên và đổi a với phần tử thứ i
    - đổi lại i với chốt sẽ được 2 nửa của chốt một bên lơn hơn một bên nhỏ hơn
    - gọi đệ quy 2 nửa còn lại*/

void Quicksort(int *a, int L, int R) //sx a[L]..a[R-1]
{
    if(L+1 >= R) return; //suy biến thì thoát (<1)
        swap(a[L], a[(L + R)/2]); //Chọn chốt là phần tử ở giữa đổi lên đầu

    int i = L; //chạy i từ L - lưu vị trí các số nhỏ hơn chốt
    for(int j = L+1; j < R; j++) //duyệt j: L+1 -> R-1
        if(a[j] < a[L]) swap(a[++i], a[j]);
            //Nếu nhỏ hơn chốt (a[L]) -> tăng i và đổi phần từ a[j] với a[i]
            swap(a[L],a[i]);//đổi lại chốt với phẩn tủ i

    Quicksort(a, L, i);//gọi đệ quy nửa trái sx L->i-1
    Quicksort(a, i+1, R);//gọi đệ quy nửa phải sx i+1->R-1
}

void quicksort(int *L, int *R)//con trỏ L đến con trỏ R-1
{
    if(L+1 >= R) return;//suy biến thì thoát
    swap(*L, *(L+(R-L)/2));//Chọn chốt là phần tử ở giữa đổi lên đầu
    int *i = L;//chạy i từ L - lưu vị trí các số nhỏ hơn chốt
    for(int *j = L+1; j < R; j++) //duyệt con trỏ j: L+1 -> R-1
        if(*j < *L)
            swap(*++i, *j);
        //Nếu nhỏ hơn chốt (*L) -> tăng i và đổi phần từ *j với *i
        swap(*L, *i);//Đổi lại chốt

    quicksort(L,i);//Gọi đệ quy nửa trái
    quicksort(i+1,R);//Gọi đệ quy nửa phải
}

template <class T,class CMP =less<T> >
void quicksort(T *L,T *R,CMP ss=less<T>() )//con trỏ L đến con trỏ R-1
{
    if(L+1 >= R) return;//suy biến thì thoát
    swap(*L, *(L+(R-L)/2));//Chọn chốt là phần tử ở giữa đổi lên đầu
    T *i = L;//chạy i từ L - lưu vị trí các số nhỏ hơn chốt
    for(T *j = L+1; j < R; j++)//duyệt con trỏ j: L+1 -> R-1
        if(ss(*j, *L)) swap(*++i, *j);
            //Nếu ss đúng -> đổi phần từ *j với *i
            swap(*L, *i);//Đổi lại chốt

    quicksort(L, i, ss);//Gọi đệ quy nửa trái
    quicksort(i+1, R, ss);//Gọi đệ quy nửa phải
}

int main()
{
    int a[] = {23, 64, 74, 68, 38, 78, 86, 62, 43, 28, 39, 18} , n = sizeof(a)/sizeof(int);
    quicksort(a, a+n, greater<int>());
    for(auto z:a) cout << z << " ";
}
```

HeapSort	Code

```
#include<bits/stdc++.h>
using namespace std;

void Heapy(int *a, int n, int k)  //vun tại vị trí k trong dãy a[0]..a[n-1]
{
    if(2*k+1 >= n) return;
    int p = 2*k+1;            //xét con 1
    if(p + 1 < n && a[p] < a[p+1]) p++; //nếu con 2 lớn hơn chuyển p sang con 2
    if(a[k] < a[p]) //nếu con lớn hơn
    {
        swap(a[k], a[p]);//hoán đổi giá trị
        Heapy(a, n, p);//vun lại tại con
    }
}
void Heapsort(int *a, int n)
{
    for(int i = n-1; i >= 0; i--) //tạo đống
        Heapy(a, n, i);
    for(int i = n - 1; i > 0; i--)
    {
        swap(a[0], a[i]);//đổi chỗ phần tử đang xét với phần tử đầu(phần tử lớn nhất)
        Heapy(a, i, 0);//vun đống lại từ vị trí 0 trong dãy a[0].. a[i-1]
    }
}

void HEAPY(int *L, int *R, int k)  //vun tại vị trí k trong day đoạn con trỏ L đến R-1
```

```
{
    if(L + 2*k+1 >= R) return;
    int *p = L + 2*k+1;           //xét con 1
    if(p+1 < R && *p < *(p+1)) p++; //nếu con 2 lớn hơn chuyển p sang con 2

    if(*(L + k) < *p) //nếu con lớn hơn
    {
        swap(*(L + k), *p);//hoán đổi giá trị
        HEAPY(L, R, p - L);//vun lại tại con
    }
}

void HEAPSORT(int *L, int *R)
{
    for(int *i = R - 1; i >= L; i--) //tạo đống
        HEAPY(L, R, i-L);
    for(int *i = R-1; i > L; i--)
    {
        swap(*L, *i);//đổi chỗ phần tử đang xét (i) với phần tử đầu(L-phần tử lớn nhất)
        HEAPY(L, i, 0);//vun đống lại từ vị trí 0 trong đoạn con trỏ từ L -> i-1
    }
}
template <class T, class CMP>
void heapy(T *L, T *R, int k, CMP ss)  //vun tại vị trí k trong day đoạn con trỏ L đến R-1
{
    if(L + 2*k+1 >= R) return;
    T *p = L + 2*k+1;           //xét con 1
    if(p+1 < R && ss(*p, *(p+1))) p++; //nếu con 2 lớn hơn chuyển p sang con 2
    if(ss(*(L+k), *p))
    {
        swap(*(L+k), *p);//hoán đổi giá trị
        heapy(L, R, p-L, ss);//vun lại tại con
    }
}
template <class T, class CMP = less<T> >
void heapsort(T *L, T *R, CMP ss = less<T>())
{
    for(T *i = R-1; i >= L; i--) //tạo đống
        heapy(L, R, i-L, ss);
    for(T *i = R-1; i > L; i--)
    {
        swap(*L, *i);//đổi chỗ phần tử đang xét (i) với phần tử đầu(L)
        heapy(L, i, 0, ss);//vun đống lại từ vị trí 0 trong đoạn con trỏ từ L -> i-1
    }
}

int main()
{
    int a[] = {53, 65, 8, 65, 435, 95, 67, 423, 71, 93, 16, 74};
    int n = sizeof(a)/sizeof(a[0]);
    heapsort(a, a+n, greater<int>());
    for(auto x:a) cout << x << " ";

}
```

Binary Search

Dãy con đơn điệu tăng dài nhất	Code

```
#include<bits/stdc++.h>
using namespace std;
/*  ý tưởng: tạo một vector thêm lần lượt các phần tử vào tạo một dãy tăng dần
    - nếu nó rỗng hoặc phần tử cuối nhỏ hơn thì thêm x vào cuối
    - ngược lại thì tìm phần tử đầu tiên >= x rồi gán nó bằng x đảm bảo cho day nhỏ nhất có
thể


    chú ý: dãy tìm ra có thể không phải dãy con tăng dài nhất, cách này chỉ tìm số phần tử
chúng không
    tim ra dãy đó*/
int main()
{
    int n, x;
    cin >> n;//nhập số phần tử
    vector <int> b;//vector chưa các phần tử trong dãy , size của vector là kết quả
    while(n--)
    {
        cin >> x;//nhập phần tử thứ i
        if(b.size() == 0 || b.back() < x)
            b.push_back(x);
        //nếu b rỗng học phần tử cuối nhỏ hơn x thì thêm x vào cuối
        else
        {
            auto p = lower_bound(b.begin(), b.end(), x);
            //tìm phần tử nhỏ hơn hoặc bằng b trong vector rồi thay thế nó bằng x;
            *p = x;
```

```
        }
    }
    cout << b.size();//kết quả
}
```

Code

```
/*  ý tưởng: ta sẽ so sánh phần tử giữa(mid) với giá trị cần tìm nếu bằng nhau là tìm thấy
ngược lại:
    - mid>x: tìm kiếm ở phía bên trái
    - mid<x: tìm kiếm ở phía bên phải
    lặp lại cho đến khi tìm ra hoặc chỉ còn 1 phần tử*/

#include<bits/stdc++.h>
using namespace std;

template <class T>
T *binary_search(T *L,T *R,T x) //dùng vòng lặp
{
    while(L + 1 <= R)//Nếu 1 phần tử thì thoát do không có trong mảng
    {
        T *M = L + (R-L)/2;//phần tử ở giữa
        if(*M == x) return M;//so sánh phần tử giữa, nếu tìm thấy trả về con trỏ
        *M < x ? L = M+1 : R = M-1;
        //nếu *M<x thì L=M+1 (tìm kiếm bên phải) ngược lại R=M-1 (tìm kiếm bên trái)
    }
    return NULL;
}

template <class T>
T *bs(T *L, T *R, T x) //dùng đệ quy - trả về con trỏ
{
    if(L+1 >= R) return NULL;//Nếu 1 phần tử trả về NULL - do khác giá trị cần tim
    T *M = L + (R-L)/2;//Phần tử giữa

    if(*M == x) return M;//Nếu bằng thì trả về
    if(*M < x) return bs(M+1,R,x);//Gọi về nưa bên phải (L=M+1) nếu giá trị phần tử giữa < x
    return bs(L,M-1,x); //ngược lại gọi về nửa bên trái (R=M-1)
}

int main()
{
    int a[] = {5, 8, 13, 16, 16, 16, 18, 18, 22 ,45, 58, 61};// dãy đã sắp xếp tăng
    int n = sizeof(a)/sizeof(a[0]);
    int x; cin>>x;
    int *p = binary_search(a, a+n, x);

    if(!p) cout << "khong co";
    else cout << "co tai vi tri " << p-a;
}
```

Các kiến thức khác

Cấu trúc tự trỏ | Code

```
#include<bits/stdc++.h>
using namespace std;
struct nguoi{
    string ten;
    int tuoi;
    struct nguoi *bo,*me;
    nguoi(string t = "vo danh", int tt = 18)//*cấu trúc tự trỏ giống như hàm tạo
    {
        ten = t;
        tuoi = tt;
        bo = me = NULL;
    }
};
int main()
{
    nguoi X("chi pheo", 15), Y("thi no") ,*Z = new nguoi("ba kien", 12);
    Z->bo = &X;
    Z->me = &Y;
    cout<<"\nX: " << X.ten << " " << X.tuoi;
    cout<<"\nY: " << Y.ten << " " << Y.tuoi;
    cout<<"\nZ: " << Z->ten << " " << (*Z).tuoi;
    cout<<"\nBo cua Z: "<< Z->bo->ten;
    cout<<"\nMe cua Z: "<< Z->me->ten;
    delete Z;
}
```

Cài linked list bằng cấu trúc tự trỏ | Code

```
#include<bits/stdc++.h>
```

```cpp
using namespace std;
struct node
{
    int elem;
    node *next;
    node(int e = 0, node *N = NULL) //cấu trúc tự trỏ giống như hàm tạo
    {
        elem = e;
        next = N;
    }
};

int main()
{
    node E, D(7,&E),*C = new node(6, &D), *B = new node(4,C), A(3,B);
    //A->*B->*C->D->E
    for(node *p = &A; p != NULL; p = p->next)
        cout << p->elem << " ";
    delete C;
    delete B;
}
```

Toán tử copy, toán tử gán	Code

```cpp
#include<bits/stdc++.h>
using namespace std;
struct ps
{
    int t,m;
    ps(int a = 0, int b = 1) {t = a; m = b;}
    ps(ps &p)
    {
        cout << "toan tu copy\n";
        t = p.t;
        m = p.m;
    }

    ps &operator=(ps &p)
    {
        cout << "toan tu assignment\n";
        this->t = p.t;
        this->m = p.m;
        return *this;
    }
};
int main()
{
    ps A(3,4);
    ps B = A;  //Su dung toan tu copy
    ps C;
    C = A;     //toan tu =
}
```

Thực hành đọc file	**Có 2 file:** **+ readfile.cpp** **+ quanly.cpp**

- readfile.cpp:

```cpp
#include<bits/stdc++.h>
using namespace std;

// Cách 1
//int main()
//{
//  vector<sv> A;
//  ifstream fin("sv.txt");
//  int n;
//  sv s;
//  fin>>n;
//  fin.ignore(1);
//  while(n--)
    // {
    //  fin>>s;
    //  A.push_back(s);
    // }
//  fin.close();
//  for(auto a:A)
    // cout << a << "\n";
//}

// Cách 2
//int main()
//{
```

```
//  FILE *f = fopen("sv.txt","r");
//  int n;
//  vector <sv> C;
//  fscanf(f, "%d\n", &n);
//  for(int i = 0; i < n; i++)
//  {
//      char ten[30];
//      int tuoi,ma;
//      double diem;
//      fscanf(f, "%[^0-9]", ten);
//      fscanf(f, "%d%d%lf\n", &ma, &tuoi, &diem);
//      sv s;
//      s.ten = ten; s.ten.pop_back();
//      s.tuoi = tuoi;
//      s.diem = diem;
//      C.push_back(s);
//  }
//  fclose(f);
//  for(auto v:C)
    // cout << setw(20) << left << v.ten <<
    //  " " << v.tuoi << " " << v.diem << "\n";
//}

//Cách 3
struct sv
{
    string ten;
    int tuoi,ma;
    double diem;
};
istream &operator >> (istream &is, sv &S)
{
    string s,x; getline(is, s);
    vector <string> z;
    istringstream ssin(s);

    while(ssin>>x)
        z.push_back(x);

    x = z.back();
    z.pop_back();
    istringstream ssin1(x);
    ssin1 >> S.diem;

    x = z.back();
    z.pop_back();
    istringstream ssin2(x);
    ssin2 >> S.tuoi;

    x = z.back();
    z.pop_back();
    istringstream ssin3(x);
    ssin3 >> S.ma;

    S.ten = "";
    for(auto x : z) S.ten += x +" ";
    if (S.ten.back() == ' ')
        S.ten.pop_back();
    return is;
}

ostream &operator<<(ostream &os, sv S)
{
    os << setw(20) << left << S.ten << " " << setw(5) << S.ma
    << " " << setw(5) << S.tuoi << " " << S.diem;
    return os;
}
```

- quanly.cpp:

```
#include<bits/stdc++.h>
#include"readfile.cpp"
using namespace std;
class quanly
{
    vector<sv> A;

    public:
        int menu()
        {
            system("cls");
            cout << "0. Nhap danh sach sv tu file\n";
            cout << "1. Them sinh vien vao cuoi\n";
            cout << "2. sap xep\n";
```

```
        cout << "3. Thay the tai vi tri k\n";
        cout << "4. Dao day nguoc lai\n";
        cout << "5. Xoa tai 1 vi tri\n";
        cout << "6. Liet ke cac phan tu:\n";
        cout << "7. Tim sinh vien co diem lon nhat\n";
        cout << "8. Liet ke nhung vi tri co max\n";
        cout << "9. Thoat";
        cout << "\nMoi ban chon : ";

        int chon; cin >> chon;
        if(0 <= chon && chon <= 9) return chon;
        return menu();
    }
    void run()
    {
        string fname;
        ifstream fin;
        int n;
        sv x;
        while(1)
        {
            switch(menu())
            {
                case 0:
                    cout<<"\nNhap ten file : ";
                    cin.ignore(1);
                    getline(cin, fname);

                    fin.open(fname,ifstream::in);
                    fin >> n;
                    fin.ignore(1);
                    while(n--)
                    {
                        fin>>x;
                        A.push_back(x);
                    }

                    fin.close();
                    break;

                case 6:
                    cout << "\nNoi dung danh sach : \n";
                    for(auto x : A)
                        cout << x << "\n";
                    break;
                default: return ;
            }
            system("pause");
        }
    }
};
int main()
{
    quanly Q;
    Q.run();
}
```

Ước chung lớn nhất

```
#include<bits/stdc++.h>
using namespace std;
/*
Note : a là số bị chia, b là số chia
giải thuật euclid :
B1 : Lấy số a chia dư cho b rồi gán số dư vào r
B2 : Nếu số dư khác 0 ( r != 0) thì ta tiếp tục lấy số chia (b) chia dư cho số dư (r) lặp
lại cho đến khi số dư bằng 0 (r = 0)
B3 : Số chia trong phép chia cuối cùng là ước chung lớn nhất của 2 số
*/
template <class T,class K> // Khuôn mẫu hàm
T UCLN(T a,K b) {
    while(b != 0) { // lặp lại chừng nào phần dư còn khác 0
        T r = a%b;
        // gán số chia cho a , số dư cho b lặp lại cho đến khi số dư bằng 0
        a = b;
        b = r;

    }
    return a;
}
int main() {
```

```
    long long a, b;
    cin >> a >> b;
    cout << UCLN<long long,long long>(a,b);
}
```

