Kiểm kê 2

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

void merge(vector<string> &a, int l, int m, int r) {
    vector<string> left(a.begin() + l, a.begin() + m + 1);
    vector<string> right(a.begin() + m + 1, a.begin() + r + 1);

    int i = 0, j = 0, k = l;
    while (i < left.size() && j < right.size()) {
        if (left[i].size() < right[j].size() ||
           (left[i].size() == right[j].size() && left[i] < right[j])) {
            a[k++] = left[i++];
        } else {
            a[k++] = right[j++];
        }
    }

    while (i < left.size()) a[k++] = left[i++];
    while (j < right.size()) a[k++] = right[j++];
}

void mergeSort(vector<string> &a, int l, int r) {
    if (l >= r) return;
    int m = (l + r) / 2;
    mergeSort(a, l, m);
    mergeSort(a, m + 1, r);
    merge(a, l, m, r);
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
        freopen("TASK.INP", "r", stdin);
        freopen("TASK.OUT", "w", stdout);
    }

    int n;
    cin >> n;
    vector<string> s(n + 1);
    for (int i = 1; i <= n; i++) {
        cin >> s[i];
    }

    mergeSort(s, 1, n);

    vector<vector<string>> adj(n + 1);

    int i = 1;
    while (i <= n) {
        int j = i;
        while (j <= n && s[i] == s[j])
            j++;

        adj[j - i].push_back(s[i]);
        i = j;
    }

    for (int i = n; i >= 0; i--) {
        if (adj[i].empty()) continue;
        for (auto res : adj[i]) {
            cout << res << ' ' << i << '\n';
        }
    }
}
```

Binary Search 2

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

struct A {
    int x, y;
};

bool cmp(A a, A b) {
    if (a.x == b.x) {
        return a.y < b.y;
    }
    return a.x < b.x;
}

bool cmp2(A a, A b) {
    if (a.x == b.x) {
        return a.y > b.y;
    }
    return a.x > b.x;
}

void merge(vector<A> &v, int l, int m, int r) {
    vector<A> left(v.begin() + l, v.begin() + m + 1);
    vector<A> right(v.begin() + m + 1, v.begin() + r + 1);
    int i = 0, j = 0, k = l;
    while (i < left.size() && j < right.size()) {
        if (cmp(left[i], right[j])) {
            v[k++] = left[i++];
        } else {
            v[k++] = right[j++];
        }
    }
    while (i < left.size()) v[k++] = left[i++];
    while (j < right.size()) v[k++] = right[j++];
}

void mergeSort(vector<A> &v, int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort(v, l, m);
        mergeSort(v, m + 1, r);
        merge(v, l, m, r);
    }
}

void merge2(vector<A> &v, int l, int m, int r) {
    vector<A> left(v.begin() + l, v.begin() + m + 1);
    vector<A> right(v.begin() + m + 1, v.begin() + r + 1);
    int i = 0, j = 0, k = l;
    while (i < left.size() && j < right.size()) {
        if (cmp2(left[i], right[j])) {
            v[k++] = left[i++];
        } else {
            v[k++] = right[j++];
        }
    }
    while (i < left.size()) v[k++] = left[i++];
    while (j < right.size()) v[k++] = right[j++];
}
```

```
void mergeSort2(vector<A> &v, int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
        mergeSort2(v, l, m);
        mergeSort2(v, m + 1, r);
        merge2(v, l, m, r);
    }
}

signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, q;
    cin >> n >> q;
    vector<A> v;
    vector<A> v2;
    for (int i=1; i<=n; i++){
        int x;
        cin >> x;
        v.push_back({x, i});
        v2.push_back({x, i});
    }
    mergeSort(v, 0, n - 1);
    mergeSort2(v2, 0, n - 1);

    while (q--){
        string s;
        cin >> s;
        int type;
        cin >> type;
        int x;
        cin >> x;
        if (type == 1){
            auto it = lower_bound(v.begin(), v.end(), A{x, 0}, cmp);
            if (it->x != x){
                cout << "-1\n";
            }
            else{
                cout << it->y << "\n";
            }
        }
        else {
            auto it = lower_bound(v2.begin(), v2.end(), A{x, 9999999}, cmp2);
            if (it->x != x){
                cout << "-1\n";
            }
            else{
                cout << it->y << "\n";
            }
        }
    }

}
```

khangtd.DetectVirusin2D

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;

class AhoCorasick {
    private:
        struct Node {
            unordered_map<char, Node*> children;
            Node* fail = nullptr;
            vector<string> outputs;
        };

        Node* root;

    public:
        AhoCorasick() {
            root = new Node();
        }

    void addWord(const string& word) {
        Node* node = root;
        for (char ch : word) {
            if (!node->children.count(ch))
                node->children[ch] = new Node();
            node = node->children[ch];
        }
        node->outputs.push_back(word);
    }
    void build() {
        queue<Node*> q;
        root->fail = root;
        for (auto& [ch, node] : root->children) {
            node->fail = root;
            q.push(node);
        }

        while (!q.empty()) {
            Node* current = q.front(); q.pop();
            for (auto& [ch, child] : current->children) {
                Node* fallback = current->fail;
                while (fallback != root && !fallback->children.count(ch)) {
                    fallback = fallback->fail;
                }
                if (fallback->children.count(ch) && fallback->children[ch] != child) {
                    child->fail = fallback->children[ch];
                } else {
                    child->fail = root;
                }
                child->outputs.insert(child->outputs.end(),
                                      child->fail->outputs.begin(),
                                      child->fail->outputs.end());
                q.push(child);
            }
        }
    }

    vector<pair<int, string>> search(const string& text) {
        vector<pair<int, string>> results;
        Node* node = root;

        for (int i = 0; i < text.size(); ++i) {
            char ch = text[i];
            while (node != root && !node->children.count(ch)) {
                node = node->fail;
            }
            if (node->children.count(ch)) {
                node = node->children[ch];
            }
            for (const string& match : node->outputs) {
                results.emplace_back(i - match.size() + 1, match);
            }
        }
        return results;
    }

    ~AhoCorasick() {
        destroy(root);
    }

private:
    void destroy(Node* node) {
        for (auto& [ch, child] : node->children) {
            destroy(child);
        }
        delete node;
    }
};


signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m, q;
    cin >> n >> m >> q;
    vector<vector<char>> a(n + 5, vector<char>(m + 5));
    for (int i=0; i<n; i++){
        for (int j=0; j<m; j++){
            cin >> a[i][j];
        }
    }
    vector<string> patt;
    while (q--){
        string x;
        cin >> x;
        patt.push_back(x);
    }
    string s1 = "", s2 = "";
    for (int i=0; i<n; i++){
        string s = "";
        for (int j=0; j<m; j++){
            s += a[i][j];
        }
        s1 += s + "$hehe$";
    }
    for (int i=0; i<m; i++){
        string s = "";
        for (int j=0; j<n; j++){
            s += a[j][i];
        }
        s2 += s + "$hehe$";
    }
    AhoCorasick ac1;
    AhoCorasick ac2;
    for (const auto& p : patt) {
        ac1.addWord(p);
        ac2.addWord(p);
    }
    ac1.build();
    ac2.build();
    map<string, int> mp;

    auto matches1 = ac1.search(s1);
    for (auto& [pos, pattern] : matches1) {
        mp[pattern]=1;
    }

    auto matches2 = ac2.search(s2);
    for (auto& [pos, pattern] : matches2) {
        mp[pattern]=1;
    }
    for (auto it : patt){
        if (mp[it]){
            cout << 1;
        }
        else cout << 0;
    }
}
```

khangtd.XepHang2

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;



signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}
    int n, q;
    cin >> n >> q;

    list<int> l;
    map<int, list<int>::iterator> pos;

    for (int i = 1; i <= n; ++i) {
        l.push_back(i);
    }

    auto it = l.begin();
    for (int i = 1; i <= n; ++i, ++it) {
        pos[i] = it;
    }

    while (q--) {
        int x;
        cin >> x;
        l.erase(pos[x]);
        l.push_front(x);
        pos[x] = l.begin();
        cout << l.back() << " ";
    }
}
```

khangtd.XepHang

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
#define fi first
#define se second
const int N = 1e6 + 9;
const int N2 = N * 10;
const int mod = 1e9 + 7;
const int inf = LLONG_MAX;



signed main(){
    ios::sync_with_stdio(false);
    cin.tie(NULL);
    if (fopen("TASK.INP", "r")){
    freopen("TASK.INP", "r", stdin);
    freopen("TASK.OUT", "w", stdout);}

    int n, m;
    cin >> n >> m;
    stack<int> st;
    map<int, int> mp;
    for (int i=n; i>=1; i--){
        st.push(i);
    }
    for (int i=0; i<m; i++){
        int x;
        cin >> x;
        st.push(x);
    }
    while (!st.empty()){
        if (!mp[st.top()]){
            cout << st.top() << " ";
            mp[st.top()] = 1;
        }
        st.pop();
    }

}
```