Coding Problems
Problem A

We have a **DailyBot** that sends everyone messages from time to time in a day in the **given order**. It has a system to display the message recipients list & at the start of a day the list is empty. Whenever it sends a message to someone his/her name pops up on **top** of the list.

Everyday, it sends a lot of messages & it's very hard to check the correctness of the display system. So we need your help to find out this list after the DailyBot has finished sending all the messages.

Input:
Input contains the name of recipients. (a person may get messages several times in a day)
You may assume that the DailyBot can handle upto a **million** messages in a day.

Output:
Output should contain the name of the message recipients & the number of times each of them received messages from the DailyBot in the order of the system described.

Sample Input	**Sample Output**
Fahim Tamal Emruz Emruz Kamol Fahim Emruz Emruz Fahim Tamal	Tamal 2 Fahim 3 Emruz 4 Kamol 1

```cpp
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
using namespace std;
typedef long long ll;

int main()
{
    vector<string> v, ans;
    map<string, ll> mp;
    string s;
    while (cin >> s)
    {
        v.push_back(s);
    }
    ll v_len = v.size();
    for (ll i = v_len - 1; i >= 0; i--)
    {
        if (!mp[v[i]])
            ans.push_back(v[i]);
        mp[v[i]]++;
    }
    ll len = ans.size();
    for (ll i = 0; i < len; i++)
    {
        cout << ans[i] << " " << mp[ans[i]] << endl;
    }
    return 0;
}
```

Problem B

We've designed a Tweet ranking algorithm & need you to implement that for us. Each tweet has **at least three** different components, the **username** of this tweet, number of **followers** of this user & the **tweet content** itself. Additionally, it may contain a **fourth** component (a **hashtag**) which indicates that this tweet is based on some "trending" topics.

So, a tweet may look like this:

Username NumberOfFollowers <TweetContent>

Or

Username NumberOfFollowers <TweetContent> #

Our tweet **ranking algorithm** is based on the following rules (**in the given order**):

1. Trending Topic
2. Higher Value of the Tweet Content
3. Most Number of Followers
4. Lexicographic order of the Username

Value of a Tweet is calculated by the given formula:
Length of the Tweet Content **x** Number of unique letters in the Tweet Content.

Input:
Input contains several lines, where each line contains a tweet in the format described above. You can safely assume that the Usernames are unique in each Tweet.

Output:
Output should contain the **Usernames, Followers** according to the ranking algorithm described above.

Sample Input	**Sample Output**
Kube 10 <Hello World> Dock 5 <Tweeting is Fun> # Gola 20 <Hey, This is the TweetContent>	Dock 5 Gola 20 Kube 10

```
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <sstream>
using namespace std;
typedef long long ll;

int main()
{
   string username;
   ll followers;
   vector<pair<pair<ll, ll>, pair<ll, string>>> ans;
   while (cin >> username)
   {
      cin >> followers;
      ll is_hash = 0;
      char ch;
      map<char, ll> mp;
      ll ch_ct = 0, uni_ch_ct = 0;
      string line;
      getline(cin, line);
      istringstream is(line);
      while (is >> ch)
      {
         if (ch == '#')
            is_hash = 1;
         else if ((ch >= 'a' && ch <= 'z') || (ch >= 'A' && ch <= 'Z'))
         {
            if (ch >= 'a' && ch <= 'z')
            {
               ch = ch - 'a' + 'A';
            }
            if (!mp[ch])
               uni_ch_ct++;
            mp[ch]++;
            ch_ct++;
         }
      }
      ll rnk = ch_ct * uni_ch_ct;
      ans.push_back({{-is_hash, -rnk}, {-followers, username}});
   }
   sort(ans.begin(), ans.end());
   ll len = ans.size();
   for (ll i = 0; i < len; i++)
```

```
    {
        cout << ans[i].second.second << " " << -ans[i].second.first << endl;
    }
    return 0;
}
```

Problem C

We have a product with a list of features which is currently in version:1 . We can only modify existing features or add a new feature to the current version & release a new version.

For example:
[A, B, C] version:1
[A, B, C, D] version:2 (we added a new feature D)
[A, B, C2, D] version:3 (we modified the feature C)

So, your code must be able to handle these 3 operations:
i. **add -** feature_name (add a new feature to current version)
ii. **modify -** existing_feature_name new_feature_name (modify an existing feature)
iii. **check -** version feature_name (check whether "feature_name" is available in this "version" or not, print "yes"/"no")

Sample Input:

[A, B, C]
Add D
Check version:2 D
Check version:2 E
Add E
Check version:3 E
Modify D D2
Check version:4 D
Check version:4 D2

Expected Output:

Yes
No
Yes
No
Yes

Explanation:

version:1 [A, B, C]
version:2 [A, B, C, D]
version:3 [A, B, C, D, E]
version:4 [A, B, C, D2, E]

```
#include <iostream>
#include <vector>
#include <map>
#include <algorithm>
#include <sstream>
using namespace std;
typedef long long ll;

int main()
{
    string line;
    getline(cin, line);
    istringstream is(line);
    vector<vector<string>> v;
    vector<string> tmp;
    string s = "";
    char ch;
    while (is >> ch)
    {
        if (ch == ',' || ch == ']')
        {
            tmp.push_back(s);
            s = "";
        }
        else
        {
            s += ch;
        }
    }
    v.push_back(tmp);
    string typ;
    while (cin >> typ)
    {
        if (typ == "Add")
        {
            string val;
            cin >> val;
            tmp = v[v.size() - 1];
            tmp.push_back(val);
            v.push_back(tmp);
        }
        else if (typ == "Check")
        {
            for (ll i = 0; i < 8; i++)
                cin >> ch;
```

```
            int vers;
            cin >> vers;
            string val;
            cin >> val;
            int len = v.size();
            bool ck = true;
            if (vers > len || vers <= 0)
                ck = false;
            else
            {
                tmp = v[vers - 1];
                ck = false;
                for (ll i = 0; i < (ll)tmp.size(); i++)
                {
                    if (tmp[i] == val)
                    {
                        ck = true;
                        break;
                    }
                }
            }
            if (ck)
                cout << "Yes\n";
            else
                cout << "No\n";
        }
        else
        {
            string o, n;
            cin >> o >> n;
            tmp = v[v.size() - 1];
            for (ll i = 0; i < (ll)tmp.size(); i++)
            {
                if (tmp[i] == o)
                {
                    tmp[i] = n;
                    break;
                }
            }
            v.push_back(tmp);
        }
    }
    return 0;

}
```