字符串摘要题目描述：

给定一个字符串的摘要算法，请输出给定字符串的摘要值。

1、去除字符串中非字母的符号。

2、如果出现连续字符（不区分大小写），则输出：该字符（小写）+ 连续出现的次数。

3、如果是非连续的字符（不区分大小写），则输出：该字符（小写）+ 该字母之后字符串中出现的该字符的次数。

4、对按照以上方式表示后的字符串进行排序：字母和紧随的数字作为一组进行排序，数字大的在前，数字相同的，则按字母进行排序，字母小的在前。

输入描述：

一行字符串，长度为[1,200]

输出描述：

摘要字符串

补充说明：

示例 1

输入：

```
aabbcc
```

输出：

```
a2b2c2
```

说明：

示例 2

输入：

```
bAaAcBb
```

输出：

```
a3b2b2c0
```

说明：

bAaAcBb：

第一个 b 非连续字母，该字母之后字符串中还出现了 2 次（最后的两个 Bb），所以输出 b2，

a 连续出现 3 次，输出 a3，

c 非连续，该字母之后字符串再没有出现过 c，输出 c0

Bb 连续 2 次，输出 b2

对 b2a3c0b2 进行排序，最终输出 a3b2b2c0

```
#include <cctype>
#include <iostream>
#include <string>
#include <vector>
#include <map>
#include <algorithm>
using namespace std;
struct node {
	char c;
	int times;
	node* next;

	bool operator <= (const node& right) {
		if (this->times > right.times) {
			return true;
		} else if (this->times < right.times) {
			return false;
		} else {
			if (this->c <= right.c) {
				return true;
			} else {
				return false;
			}

		}

	}
	bool operator == (const node& right) {
		return this->times == right.times && this->c == right.c;
```

```
        }

        bool operator < (const node& right) {
            if (*this == right) {
                return false;
            }
            return *this <= right;
        }

};

int main() {
        string input;
        getline(cin, input);
        vector<node> ret;
        map<char, node> prev;
        int times = 1;
        string s;
        for(char i : input){
            if (isalpha(i)){
                s.push_back(i);
            }
        }
        for (int i = s.length() - 1; i >= 0 ; i--) {
            if (!(isalpha(s[i]))) {

                continue;
            } else if (i != 0 && tolower(s[i]) == tolower(s[i - 1])) {
                times ++;
                continue;
            }

            char cur = (char)tolower(s[i]);
            if (times == 1) {
                if (prev.find(cur) == prev.end()) {
                    times = 0;
                } else {

                    times = prev[cur].times;
                }
            }

            node n = {.c = cur, .times = times};
            times = 1;
```

```
            prev.insert(std::pair<char, node>(cur, n));
            ret.push_back(n);

        }
        sort(ret.begin(), ret.end());
        for (auto& j : ret) {
            cout << j.c << j.times;
        }
        cout << endl;
}
// 64 位输出请用 printf("%lld")
```