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

```
import java.util.*;
import java.util.function.Function;
import java.util.stream.Collectors;
import java.util.stream.IntStream;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        while (sc.hasNext()) {
            String str = sc.next().toLowerCase();
            String res = cal(str);
            System.out.println(res);
        }
```

```
    }

    private static String cal(String str) {
        String temp = "";
        for (int i = 0; i < str.length(); i++) {
            if (Character.isLetter(str.charAt(i))) {
                temp += str.charAt(i);
            }
        }
        if (temp == null || temp.equals("")) {
            return "";
        }
        Map<Character, Long> eleCountMap = IntStream.range(0, temp.length())
                .mapToObj(temp::charAt)
                .collect(Collectors.groupingBy(Function.identity(), Collectors.counting()));
        int tempCount = 1;
        List<Node> res = new ArrayList<>();
        char pre = temp.charAt(0);
        eleCountMap.put(pre, eleCountMap.get(pre) - 1);
        for (int i = 1; i < temp.length(); i++) {
            char cur = temp.charAt(i);
            eleCountMap.put(cur, eleCountMap.get(cur) - 1);
            if (cur == pre) {
                tempCount++;
            } else {
                res.add(new Node(pre, tempCount > 1 ? tempCount :
eleCountMap.get(pre)));
                pre = cur;
                tempCount = 1;
            }
        }
        res.add(new Node(pre, tempCount > 1 ? tempCount : eleCountMap.get(pre)));
        return res.stream().sorted((a, b) -> {
            if (a.count == b.count) {
                return a.ele - b.ele;
            }
            return (int) (b.count - a.count);
        }).map(a -> a.ele + "" + a.count).collect(Collectors.joining());
    }

    static class Node {
        char ele;
        long count;
```

```
        public Node(char ele, long count) {
            this.ele = ele;
            this.count = count;
        }
    }
}
```