服务失效判断

题目描述：

某系统中有众多服务，每个服务用字符串（只包含字母和数字，长度<=10）唯一标识，服务间可能有依赖关系，如 A 依赖 B，则当 B 故障时导致 A 也故障。

依赖具有传递性，如 A 依赖 B，B 依赖 C，当 C 故障时导致 B 故障，也导致 A 故障。

给出所有依赖关系，以及当前已知故障服务，要求输出所有正常服务。

依赖关系：服务 1-服务 2 表示“服务 1”依赖“服务 2”

不必考虑输入异常，用例保证：依赖关系列表、故障列表非空，且依赖关系数，故障服务数都不会超过 3000，服务标识格式正常。

输入描述：

半角逗号分隔的依赖关系列表（换行）

半角逗号分隔的故障服务列表

输出描述：

依赖关系列表中提及的所有服务中可以正常工作的服务列表，用半角逗号分隔，按依赖关系列表中出现的次序排序。

特别的，没有正常节点输出单独一个半角逗号。

示例 1

输入：

```
a1-a2,a5-a6,a2-a3

a5,a2
```

输出：

```
a6,a3
```

说明：

a1 依赖 a2，a2 依赖 a3，所以 a2 故障，导致 a1 不可用，但不影响 a3；a5 故障不影响 a6。所以可用的是 a3、a6，在依赖关系列表中 a6 先出现，所以输出:a6,a3

示例 2

输入：

```
a1-a2
a2
```

输出：

```
,
```

说明：

a1 依赖 a2，a2 故障导致 a1 也故障，没有正常节点，输出一个逗号

```
import java.util.*;


// 注意类名必须为 Main, 不要有任何 package xxx 信息
public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        // 注意 hasNext 和 hasNextLine 的区别
        String[] s1 = in.nextLine().split(",");
        String[] s2 = in.nextLine().split(",");
        HashMap<String, LinkedList<String>> hm = new HashMap<>();
        HashMap<String, Boolean> wrong = new HashMap<>();
        LinkedList<String> list1 = new LinkedList<>();
        LinkedList<String> list2 = new LinkedList<>();
        for (String s : s1) {
```

```
                String[] tem = s.split("-");
                list1.add(tem[0]);
                list2.add(tem[1]);
                wrong.put(tem[0], false);
                wrong.put(tem[1], false);
                if (!hm.containsKey(tem[1])) hm.put(tem[1], new
LinkedList<String>());
                hm.get(tem[1]).add(tem[0]);
            }
            Deque<String> dq = new LinkedList<>();
            for (String s : s2) {
                if (!wrong.get(s)) {
                    wrong.put(s, true);
                    dq.offerLast(s);
                    while (!dq.isEmpty()) {
                        String tt = dq.pollFirst();
                        if (hm.containsKey(tt)) {
                            for (String ch : hm.get(tt)) {
                                if (!wrong.get(ch)) {
                                    wrong.put(ch, true);
                                    dq.offerLast(ch);
                                }
```

```
                }
            }


        }
    }
}


    boolean f = false;
    for (int i = 0; i < list1.size(); i++) {
        if (!wrong.get(list1.get(i))) {
            if (f) System.out.print("," + list1.get(i));
            else {
                f = true;
                System.out.print(list1.get(i));
            }
            wrong.put(list1.get(i), true);
        }
        if (!wrong.get(list2.get(i))) {
            if (f) System.out.print("," + list2.get(i));
            else {
                f = true;
                System.out.print(list2.get(i));
```

```
                    }
                    wrong.put(list2.get(i), true);
                }
            }
            if (!f) System.out.print(',');
        }
    }
```