树状结构查询

题目描述：

通常使用多行的节点、父节点表示一棵树，比如

西安 陕西

陕西 中国

江西 中国

中国 亚洲

泰国 亚洲

输入一个节点之后，请打印出来树中他的所有下层节点

输入描述：

第一行输入行数，下面是多行数据，每行以空格区分节点和父节点

接着是查询节点

输出描述：

输出查询节点的所有下层节点。以字典序排序

补充说明：
树中的节点是唯一的，不会出现两个节点，是同一个名字

示例

示例 1
输入：

```
5
b a
c a
d c
e c
f d
c
```

输出：

```
d
e
f
```

说明：

```
import java.nio.charset.StandardCharsets;
import java.util.*;

public class Main {

    public static class node1 {
        String name;
        PriorityQueue<String> pq = new PriorityQueue<>();



        public String getName() {
            return name;
        }

        public void setName(String name) {
            this.name = name;
        }


        public node1(String name) {
            this.name = name;
        }

        public PriorityQueue<String> getPq() {
            return pq;
        }

        public void setPq(PriorityQueue<String> pq) {
            this.pq = pq;
        }
    }




    public static void main(String[] args) {
        /*Scanner cin = new Scanner(System.in,
StandardCharsets.UTF_8.name());
        int num = cin.nextInt();
        int[] steps = new int[num];
        for (int i = 0; i < num; i++) {
            steps[i] = cin.nextInt();
```

```
}
cin.close();*/


HashMap<String, node1> Map = new HashMap<>();
Scanner sc = new Scanner(System.in);
int n = Integer.parseInt(sc.nextLine());
for (int i = 0; i < n; i++) {
	String s = sc.nextLine();
	String[] s1 = s.split(" ");

	if(!Map.containsKey(s1[1])) {
		Map.put(s1[1], new node1(s1[1]));
	}
	if(!Map.containsKey(s1[0])) {
		Map.put(s1[0], new node1(s1[0]));
	}

	Map.get(s1[1]).getPq().add(s1[0]);

}

String s = sc.nextLine();
PriorityQueue<String> left = new PriorityQueue<>();
left.add(s);
ArrayList<String> res = new ArrayList<>();
while (left.size() != 0) {
	String s2 = left.poll();
	res.add(s2);
	PriorityQueue<String> pq2 = Map.get(s2).pq;
	while (pq2.size() != 0) {
		left.add(pq2.poll());
	}
}
res.remove(s);

res.sort(new Comparator<String>() {
	@Override
	public int compare(String o1, String o2) {
		return o1.compareTo(o2);
	}
});

for (String re : res) {
```

```
                System.out.println(re);
            }



        }
    }
```