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 1
输入：
5
b a
c a
d c
e c
f d
c
输出：
d
e
f
说明：

```
import java.util.*;

/**
 * @作者 Brown
 * @日期 2023/6/21 16:12
 */
public class Main {
    public static int[] h, ne;
    public static String[] e;
    public static int idx = 0, cnt = 0;
    public static HashMap<String, Integer> map;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
```

```
        int n = sc.nextInt() + 1;//n 个结点
        map = new HashMap<>();
        h = new int[n];
        e = new String[n];
        ne = new int[n];
        for (int i = 0; i < n; i++) {
            h[i] = -1;
        }
        for (int i = 0; i < n - 1; i++) {//n 行数据
            String a = sc.next(), b = sc.next();
            add(b, a);//b 到 a 有条边
        }
        String st = sc.next();
        Queue<String> q = new LinkedList<>();
        q.add(st);
        ArrayList<String> res = new ArrayList<>();
        while (!q.isEmpty()) {
            String t = q.poll();
            for (int i = h[get(t)]; i != -1; i = ne[i]) {
                res.add(e[i]);
                q.add(e[i]);
            }
        }
        res.sort(String::compareTo);
        for (String s : res) {
            System.out.println(s);
        }
    }

    public static int get(String x) {
        if (!map.containsKey(x)) map.put(x, cnt++);
        return map.get(x);
    }

    public static void add(String a, String b) {
        e[idx] = b;
        ne[idx] = h[get(a)];
        h[get(a)] = idx++;
    }
}
```