

目录删除题目描述：

某文件系统中存在 N 个目录，每个目录都有一个独一无二的 ID 。每个目录只有一个父目录，但每个父目录下可以有零个或者多个子目录，目录结构呈树状结构。

假设，根目录的 ID 为 0 ，且根目录没有父目录，其他所有目录的 ID 用唯一的正整数表示，并统一编号。

现给定目录 ID 和其父目录 ID 的对应父子关系表[子目录 ID ，父目录 ID]，以及一个待删除的目录 ID ，请计算并返回一个 ID 序列，表示因为删除指定目录后剩下的所有目录，返回的 ID 序列以递增序输出。

注意：

- 1、被删除的目录或文件编号一定在输入的 ID 序列中；
- 2、当一个目录删除时，它所有的子目录都会被删除。

输入描述：

输入的第一行为父子关系表的长度 m ；接下来的 m 行为 m 个父子关系对；最后一行为待删除的 ID 。序列中的元素以空格分割，参见样例。

输出描述：

输出一个序列，表示因为删除指定目录后，剩余的目录 ID 。

补充说明：

示例 1

输入：

5
8 6
10 8
6 0
20 8
2 6
8

输出：

2 6

说明：

目录结构如下所示：

```

    6
  /  \
2      8
  /  \
10  20
```

删除目录 8，同时它的子目录 10 也被删除，剩余 2 和 6 两个目录。

```
import java.util.*;
```

```
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int m = sc.nextInt();
        int[][] relations = new int[m][2];
        for(int i=0; i<m; i++){
            relations[i][0] = sc.nextInt();
            relations[i][1] = sc.nextInt();
        }
        int del = sc.nextInt();
        System.out.println(getResult(m, relations, del));
    }

    public static String getResult(int m, int[][] relations, int del){
        HashMap<Integer, ArrayList<Integer>> tree = new HashMap<>();
        for(int[] relation: relations){
            int child = relation[0];
            int father = relation[1];
            tree.putIfAbsent(father, new ArrayList<>());
            tree.get(father).add(child);
        }
    }
}
```

```

        if (del == 0){
            return "";
        }

        ArrayList<Integer> res =new ArrayList<>();
        dfs(tree,0,del,res);

        res.sort((a,b)->a-b);
        StringJoiner sj= new StringJoiner(" ");
        for(Integer v:res){
            sj.add(v+"");
        }

        return sj.toString();
    }

    public static void dfs(HashMap<Integer,ArrayList<Integer>>tree,int node,int
    del,ArrayList<Integer>res){
        if(tree.containsKey(node)){
            ArrayList<Integer>children=tree.get(node);
            for(Integer child :children ){
                if(child != del){
                    res.add(child);
                    dfs(tree,child,del,res);

                }

            }

        }

    }

}

}

```