告警抑制题目描述：

告警抑制，是指高优先级告警抑制低优先级告警的规则。高优先级告警产生后，低优先级告警不再产生。请根据原始告警列表和告警抑制关系，给出实际产生的告警列表。

注：不会出现循环抑制的情况。

告警不会传递，比如 A->B, B->C，这种情况下 A 不会直接抑制 C。

但被抑制的告警仍然可以抑制其他低优先级告警。

输入描述：

第一行为数字 N，表示告警抑制关系个数，0 <= N <= 120

接下来 N 行，每行是由空格分隔的两个告警 ID，例如：id1 id2，表示 id1 抑制 id2，告警 ID 的格式为 大写字母+0 个或者 1 个数字

最后一行为告警产生列表，列表长度[1,100]

输出描述：

真实产生的告警列表

补充说明：

告警 ID 之间以单个空格分隔

```
const rl = require("readline").createInterface({ input: process.stdin });

var iter = rl[Symbol.asyncIterator]();

const readline = async () => (await iter.next()).value;
```

```
void async function () {
    // Write your code here
    let mixArr = []
    while(line = await readline()){
        mixArr.push(line)
    }
    const n = Number(mixArr[0])
    const initArr = mixArr[mixArr.length-1].split(' ')
    const yizhiArr = mixArr.slice(1,mixArr.length-1)
    let topArr = []
    let upperArr = []
    yizhiArr.forEach((item,index) => {
        const topItem = {
            index:index,
            value: item.split(' ')[0]
        }
        const upItem = {
            index:index,
            value: item.split(' ')[1]
        }
        topArr.push(topItem)
        upperArr.push(upItem)
```

```
    })
    let arr = []
    for(let i = 0; i<initArr.length;i++){
        const flag1 = upperArr.find(item=>item.value == initArr[i])
        if(!flag1){
            arr.push(initArr[i])
        } else {
            const index = flag1.index
            const upperItem = topArr.find(item=>item.index == index).value
            const flag2 = initArr.find(item=>item == upperItem)
            if(!flag2) {
                arr.push(initArr[i])
            }
        }
    }
    console.log(arr.join(' '))
}()
```