题目描述：对报文进行重传和重排序是常用的可靠性机制，重传缓冲区内有一定数量的子报文，每个子报文在原始报文中的顺序已知，现在需要恢复出原始报文。

输入描述：输入第一行为N，表示子报文的个数，$0 < N \leq 1000$。
输入第二行为N个子报文，以空格分开，子报文格式为字符串报文内容+后缀顺序索引，字符串报文内容由[a-z,A-Z]组成，后缀为整形值，表示顺序。顺序值唯一，不重复。

输出描述：输出恢复出的原始报文。按照每个子报文的顺序的升序排序恢复出原始报文，顺序后缀需要从恢复出的报文中删除掉。

补充说明：

示例 1

输入：

```
4

rolling3 stone4 like1 a2
```

输出：

```
like a rolling stone
```

说明：

4 个子报文的内容分别为 'rolling', 'stone', 'like', 'a'，顺序值分别为 3，4，1，2，按照顺序值升序并删除掉顺序后缀，得到恢复的原始报文：like a rolling stone

示例 2

输入：

```
8
gifts6 and7  Exchanging1 all2 precious5 things8 kinds3 of4
```

输出：

```
Exchanging all kinds of precious gifts and things
```

说明：

```
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
        while (in.hasNext()) { // 注意 while 处理多个 case
            int num = in.nextInt();
```

```
                HashMap<Integer, String> map = new HashMap<>();
                for(int i = 0; i < num; i++) {
                    String s = in.next();
                    int index = 0;
                    for(int j = 0; j < s.length(); j++) {
                        if(Character.isDigit(s.charAt(j))) {
                            index = j;
                            break;
                        }
                    }
                    map.put(Integer.parseInt(s.substring(index)), s.substring(0,index));
                }

                StringBuilder sb = new StringBuilder();
                // 报文序号不一定从 1 开始，因此需要以 keyset 中的值取 value，取之前应对
key 从小到大排序
                // ArrayList<Integer> list = new ArrayList<>();
                for(int i : map.keySet()) {
                    sb.append(map.get(i)).append(" ");
                }
                System.out.println(sb.toString());
            }
        }
}
```