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：

```
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main() {
    int a;
    while (cin >> a) { // 注意 while 处理多个 case
        vector<string> strArr;
        vector<int> idxArr;
        for (int i = 0; i < a; i++) {
            string b;
            cin>>b;
            int idx = 0;
            for (int j = b.size() - 1; j >= 0; j--) {
                if (b[j] - '0' > 10) {
                    strArr.push_back(b.substr(0, j + 1));
                    idxArr.push_back(idx);
                    break;
                }
                idx += idx * 10 + (b[j] - '0');
            }
        }
        bool swapFlg = false;
        for (int i = 0; i < strArr.size(); i++) {
            for (int j = 0; j < strArr.size() - i - 1; j++) {
                if (idxArr[j] > idxArr[j + 1]) {
                    string temp = strArr[j];
                    int tempIdx = idxArr[j];
                    strArr[j] = strArr[j + 1];
                    idxArr[j] = idxArr[j + 1];
                    strArr[j + 1] = temp;
```

```
                    idxArr[j + 1] = tempIdx;
                    swapFlg = true;
                }
            }
            if (!swapFlg) break;
        }
        for (int i = 0; i < strArr.size(); i++) {
            if (i == strArr.size() - 1) {
                cout<<strArr[i];
            } else {
                cout<<strArr[i]<<" ";
            }
        }
    }
    return 0;
}
// 64 位输出请用 printf("%lld")
```