题目描述：

一个应用启动时，会有多个初始化任务需要执行，并且任务之间有依赖关系，例如 A 任务依赖 B 任务，那么必须在 B 任务执行完成之后，才能开始执行 A 任务。

现在给出多条任务依赖关系的规则，请输入任务的顺序执行序列，规则采用贪婪策略，即一个任务如果没有依赖的任务，则立刻开始执行，如果同时有多个任务要执行，则根据任务名称字母顺序排序。

例如：B 任务依赖 A 任务，C 任务依赖 A 任务，D 任务依赖 B 任务和 C 任务，同时，D 任务还依赖 E 任务。那么执行任务的顺序由先到后是：A 任务，E 任务，B 任务，C 任务，D 任务。这里 A 和 E 任务都是没有依赖的，立即执行

输入描述：

输入参数每个元素都表示任意两个任务之间的依赖关系，输入参数中符号“->”表示依赖方向，例如 A->B 表示 A 依赖 B，多个依赖之间用单个空格分割

输出描述：

输出为排序后的启动任务列表，多个任务之间用单个空格分割

补充说明：

示例 1

输入：

```
A->B C->B
```

输出：

```
B A C
```

说明：

输入参数每个元素都表示任意两个任务之间的依赖关系，输入参数中符号“ ->”表示依赖方向，例如 A->B 表示 A 依赖 B，多个依赖之间用单个空格分割，输出的多个任务之间也用单个空格分割

```
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
    while ((line = await readline())) {
        const result = [];
        const map = new Map();
        const rules = line.split(" ");

        for (let rule of rules) {
            let [dep, wasDep] = rule.split("->");
            if (map.has(dep)) {
                map.get(dep).add(wasDep);
            } else {
                map.set(dep, new Set(wasDep));
            }

            if (!map.has(wasDep)) {
                map.set(wasDep, new Set());
            }
        }

        while (!map.keys().next().done) {
            const queue = [];
            for (let item of map.keys()) {
                if (map.get(item).size === 0) {
                    queue.push(item);
                    map.delete(item);
                }
            }

            for (let item of map.keys()) {
                item;
                let wasDeps = map.get(item);

                for (var queueItem of queue) {
                    if (wasDeps.has(queueItem)) {
```

```
                                wasDeps.delete(queueItem);
                            }
                        }
                    }
                    queue.sort();
                    result.push(...queue);
                }

                console.log(result.join(" "));
            }
        })();
```