铁匠铸造

请务必看完整个文档再开始做题，前半部分对理解题目很有帮助，后半部分对拿分很有帮助

题目描述

现在有一个铁匠铺，里面有一个或多个不知疲倦的铁匠，可以不断地处理兵器订单而不休息。你需要根据给出的规则实现一个铁匠铺系统

- 铁匠的属性如下：

属性	类型	说明
uid	`int`	铁匠的 id，保证 id 不相同，且为正整数
type	`int`	铁匠本身的类型，每个铁匠只能处理与自身 `type` 一致的兵器订单

- 兵器订单的属性如下：

属性	类型	说明
oid	`int`	订单的 id，保证 id 不相同，且为正整数
priority	`int`	订单优先级，**保证所有的兵器订单优先级均不相同**，且为正整数
time	`int`	订单完成需要的时间单位，为正整数
type	`int`	订单的类型，订单只能被与自身 `type` 一致的铁匠处理

铁匠铺说明

- **铁匠铺对铁匠和订单类型的限制：**
  - 一个铁匠铺中不会出现两个 `type` 相同的铁匠。
  - 铁匠只能处理与自己 `type` 一致的兵器订单，订单也只能被与自身 `type` 一致的铁匠处理。
  - **铁匠铺接受的订单一定有与之 `type` 相同的铁匠可以处理它**，也就是说如果有 `type` 为 x 的订单，则肯定有唯一的 `type` 为 x 的铁匠。
- **铁匠铺对订单数量的限制：**
  - 铁匠铺能够暂存的兵器订单数量是**有限**的(这里的暂存订单指的是铁匠铺中尚未开始处理的订单，正在被处理、已完成或被丢弃的订单不计入暂存订单)。
  - 当某时刻来到新的订单时，如果当前暂存的订单数量已经到达上限，则丢弃一个当前店铺**所有暂存订单中**优先级最低的订单，将此时刻到来的订单加入暂存订单集合。
- **铁匠铺的调度策略如下：**
  - 在每个整数时刻到来的时候，若铁匠当前时刻在休息或者铁匠恰好在当前时刻完成手上的订单，则铁匠会尝试从暂存订单集合中选择自己能够完成（即，`type` 与自己相同）的**优先级最高**的订单开始处理，如果没有自己能完成的订单，则休息。
  - 每个铁匠同一时刻只能处理一个订单。铁匠会一直处理当前手上的订单，**直到该订单处理完成**，再尝试获取下一份订单。

输入描述

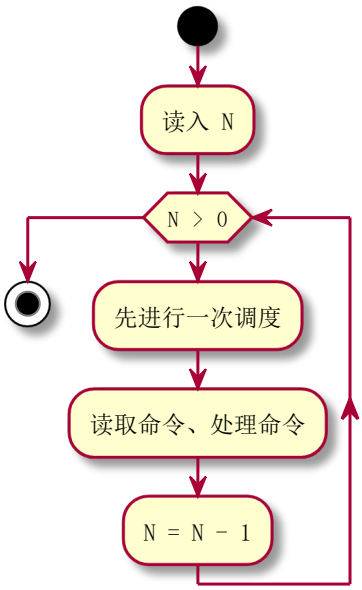
第一行为空格隔开的两个数 `[num] [limit]`,分别表示铁匠数量和铁匠铺最多允许暂存的订单的数量。

接下来有 `num` 行，表示每个铁匠的属性，每行两个数，空格隔开，分别表示 `uid` 和 `type`，**保证 `type` 互不相同**

接下来一行是一个整数 `N` ，表示有 `N` 个时间片单位，每个整数时刻都需要处理一条命令，命令格式如下：

- `add [oid] [priority] [time] [type]`：增加订单，保证每个订单一定有与之 `type` 相同的铁匠可以处理它
- `queryUser [uid]`：查询这个 `uid` 对应的铁匠当前在处理的兵器订单，输出 `worker [uid] doing order [oid]`,如果当前铁匠在休息，输出 `worker [uid] resting`，保证 `uid` 对应的铁匠一定存在
- `queryOrder [oid]`：查询这个 `oid` 对应的订单当前的状态，如果已完成，输出 `order [oid] done`,如果尚未开始处理，输出 `order [oid] pending`，如果正在被处理，输出 `order [oid] doing`，如果订单被丢弃，输出 `order [oid] discarded`，保证 `oid` 对应的订单一定存在
- `queryOrders [uid]`：按照订单完成的先后顺序查询这个 `uid` 对应的铁匠已经完成的订单集合，输出它们的 `oid`，`oid` 之间用一个空格隔开，保证 `uid` 对应的铁匠一定存在

部分流程如下，请严格按照该流程处理！



输出描述

输出每个 `query*` 的结果，每个命令的输出占一行

示例

示例 1

输入

```
1 10  // 一个铁匠，最多允许暂存的订单的数量为 10
1 1   // 铁匠的 uid 为 1，type 为 1
8     // 下面需要处理 8 个时刻的 8 条命令
queryUser 1
add 1 5 3 1  // 增加订单，oid 为 1，priority 为 5，time 为 3，type 为 1
queryUser 1
add 2 6 2 1
queryUser 1
queryUser 1
queryUser 1
queryUser 1
```

输出

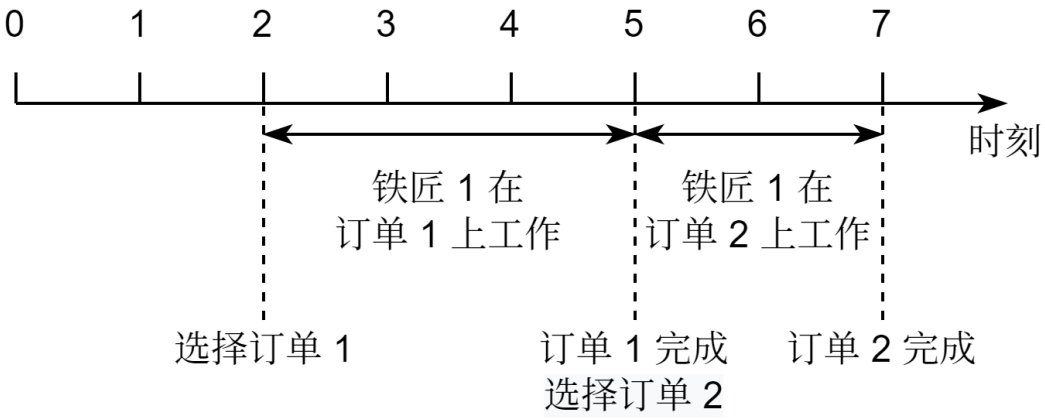
```
worker 1 resting
worker 1 doing order 1
worker 1 doing order 1
worker 1 doing order 2
worker 1 doing order 2
worker 1 resting
```

对输入输出的解释

需要处理 8 个时刻的命令，从 `0` 开始计数。

时刻	命令	备注
0	`queryUser 1`	当前还没有任何订单，因此铁匠 1 是休息状态，输出 `worker 1 resting`
1	`add 1 5 3 1`	先进行调度，而此时依然没有任何订单，所以铁匠还是休息状态；一个订单加入到任务队列中
2	`queryUser 1`	先进行调度，铁匠 1 选取 `oid` 为 1 的订单，**转为工作状态**，因而输出为 `worker 1 doing order 1`
3	`add 2 6 2 1`	先进行调度，铁匠 1 在 `oid` 为 1 的订单上进行 1 个时间片的工作，该订单还需要 2 个时间片才能完成；一个订单加入任务队列中
4	`queryUser 1`	先进行调度，铁匠 1 在 `oid` 为 1 的订单上进行 1 个时间片的工作，该订单还需要 1 个时间片才能完成，因此输出为 `worker 1 doing order 1`
5	`queryUser 1`	先进行调度，铁匠 1 在 `oid` 为 1 的订单上进行 1 个时间片的工作，`oid` 为 1 的订单完成，**立即** 选取 `oid` 为 2 的订单，铁匠依然处于工作状态，因此输出为 `worker 1 doing order 2`
6	`queryUser 1`	先进行调度，铁匠 1 在 ID 为 2 的订单上进行 1 个时间片的工作，该订单还需要 1 个时间片才能完成，因此输出为 `worker 1 doing order 2`
7	`queryUser 1`	先进行调度，铁匠 1 在 ID 为 2 的订单上进行 1 个时间片的工作，ID 为 2 的订单完成，由于没有其他订单，铁匠 1 **转为休息状态**，因此输出为 `worker 1 resting`

下图以时间轴的形式展示了铁匠 1 的活动：



示例2

输入

```
1 2
1 1
10
add 1 5 7 1
queryOrder 1
add 8 1 3 1
add 4 10 2 1
add 5 2 2 1
add 6 6 2 1
queryOrder 1
add 2 7 2 1
queryOrder 6
queryOrder 2
```

输出

```
order 1 doing
order 1 doing
order 6 discarded
order 2 pending
```

示例3

输入

```
1 10
1 1
10
add 1 5 3 1
queryOrders 1
add 2 6 2 1
queryOrders 1
queryOrders 1
queryOrders 1
queryOrders 1
queryOrders 1
queryOrders 1
queryOrders 1
```

输出

```
     // 这是一个空行，对应于第一个 queryOrders 1，由于该铁匠当前尚未完成任何订单，因此输出为空
     // 同上
1
1
1 2
1 2
1 2
1 2
```

示例4

输入

```
1 3
1 1
15
add 1 5 4 1
queryUser 1
add 2 6 8 1
queryOrder 1
add 3 2 10 1
queryOrder 3
queryOrders 1
add 6 1 5 1
queryOrder 6
add 5 10 2 1
add 100 11 2 1
add 7 12 2 1
queryOrder 6
queryUser 1
queryOrders 1
```

输出

```
worker 1 doing order 1
order 1 doing
order 3 pending
1
order 6 pending
order 6 discarded
worker 1 doing order 7
1 2
```

用例设计

#	命令组合	铁匠个数	用例百分比
1	只包含单个命令	/	4/25
2	只包含`add`、`queryUser`命令	1	2/25
3	只包含`add`、`queryUser`命令	`>1`	2/25
4	只包含`add`、`queryOrder`命令	1	2/25
5	只包含`add`、`queryOrder`命令	`>1`	2/25
6	只包含`add`、`queryOrders`命令	1	2/25
7	只包含`add`、`queryOrders`命令	`>1`	2/25
8	包含`add`、`queryUser`、`queryOrder`、`queryOrders`	1	2/25
9	包含`add`、`queryUser`、`queryOrder`、`queryOrders`	`>1`	7/25

如无法实现全部功能，请实现部分命令，近半数的用例无需考虑铁匠铺最多允许暂存的订单的数量！