Strange Regulations

`regulate.c`, `regulate.C`, `regulate.java`

Thank to cryptography, we are able to encrypt messages such that noone (except the intended recipient) is able to read them. However, encrypted messages are of no use if they do not actually *reach* the recipient. These days, computer network is the most typical mean to send such messages. In this problem, we will study the issues the networking providers have to solve. And remember: since the message is encrypted, we do not need to care about the network privacy anymore.

The network cables joining computers (servers) belong to different companies. A new anti-monopoly legislation prevents any company from owning more than two cables from each server. Furthermore, to avoid wasting resources, there is also a law specifying that the cable system owned by any single company cannot be redundant, i.e., removal of *any* of the cables will disconnect some two previously connected servers. Since the companies buy and sell the cables all the time, it is quite difficult to enforce these regulations. Your task is to write a program that does so.

Input Specification

The input contains several instances. The first line of each instance contains four integers N, M, C and T separated by spaces — the number of servers ($1 \leq N \leq 8\,000$), the number of cables ($0 \leq M \leq 100\,000$), the number of companies ($1 \leq C \leq 100$), and the number of cable-selling transactions ($0 \leq T \leq 100\,000$), respectively.

The following M lines describe the cables. Each of them contains three integers S_{j1}, S_{j2} and K_j, separated by spaces, giving the numbers of the servers S_{j1} and S_{j2} ($1 \leq S_{j1} < S_{j2} \leq n$) joined by that cable and the number of the company K_j ($1 \leq K_j \leq C$) initially owning the cable. For each pair of servers, there is at most one cable joining them. The initial state satisfies the regulations, i.e., each company owns at most two cables incident with each server, and the system of cables owned by a single company has no cycles.

Finally, each of the next T lines contains integers S_{i1}, S_{i2} and K_i describing one transaction in which the company K_i ($1 \leq K_i \leq C$) tries to buy a cable between servers S_{i1} and S_{i2} ($1 \leq S_{i1} < S_{i2} \leq N$).

The last instance is followed by a line containing four zeros.

Output Specification

For each input instance, output T lines describing the outcome of the transactions. The possible outcomes are

- "`No such cable.`" if the pair of servers is not joined by a cable,
- "`Already owned.`" if the cable is already owned by the company K_i,
- "`Forbidden: monopoly.`" if the company K_i already owns two cables at S_{i1} or S_{i2},
- "`Forbidden: redundant.`" if K_i owns at most one cable at each of S_{i1} and S_{i2}, but granting the ownership would create a cycle of cables owned by K_i,
- "`Sold.`" if none of the above restrictions apply. In this case, the ownership of the cable between S_{i1} and S_{i2} changes to K_i for the purpose of further transactions.

  Print one empty line after each instance.

Sample Input

```
4 5 3 5
1 2 1
2 3 1
3 4 2
1 4 2
1 3 3
1 2 3
1 2 3
1 4 3
2 3 3
2 4 3
2 1 1 1
1 2 1
1 2 1
0 0 0 0
```

Output for Sample Input

```
Sold.
Already owned.
Forbidden: monopoly.
Forbidden: redundant.
No such cable.

Already owned.
```