You are invited to be a part of the team that is developing yet another DBMS (Data Base Management System). You will be responsible for the Lock Manager.

Locks control concurrent access to data items by multiple transactions. Your DBMS is simple and uses only Shared (`S`) and Exclusive (`X`) mode locks. Each lock request contains a lock mode (`S` or `X`), a transaction identifier and a data item identifier. Multiple locks can be granted to the same data item as long as none of them **conflict**.

Two locks for the same data item conflict if:

- they belong to different transactions, and
- at least one of them is exclusive (`X`) mode lock.

At the earliest stages of development you are asked to write very simple lock manager that processes lock requests. The lock is granted if it does not conflict with previously granted locks for this data item. Your task is simple: locks, once granted, are never released or changed in any way. If lock request is denied due to conflict with some previously granted lock, then transaction making this request is blocked and all further requests from this transaction are ignored.

Input

The first line of the input consists of an integer indicating the number of test cases in the input. Then there is a blank line and the test cases separated by a blank line.

Each test case consists of a number of lock requests, each request on a different line. Requests have the following format:

MODE TRID ITEM

Where *MODE* is a single capital letter '`S`' or '`X`' denoting requested lock mode. *TRID* and *ITEM* are transaction identifier and data item identifier correspondingly. Both *TRID* and *ITEM* are integers, both are greater than zero, and both consist of at most 9 decimal digits.

There are at least one and at most 10000 requests in each test case.

The last request is followed by a line consisting of a single character '`#`'.

Output

Your program shall sequentially process all requests from from each test case. For each request you should write one line that contains the response to the request. The following responses are allowed:

- `GRANTED` — the lock request does not conflict with any previously granted locks and is granted.
- `DENIED` — the lock request conflicts with some previously granted lock and is denied, thus blocking the requesting transaction.
- `IGNORED` — the transaction was blocked on some request before this one.

Responses shall appear in all capital letters exactly as shown above.

Print a blank line between consecutive cases.

Sample Input

```
1

S 1 1
S 2 2
X 10 1
S 6 123456789
S 3 3
X 2 2
S 5 6
S 3 1
S 3 2
X 987654321 123456789
X 1 4
S 6 6
S 3 5
S 2 4
X 4 5
S 2 51
#
```

Sample Output

```
GRANTED
GRANTED
DENIED
GRANTED
GRANTED
GRANTED
GRANTED
GRANTED
DENIED
DENIED
GRANTED
GRANTED
IGNORED
DENIED
GRANTED
IGNORED
```