540 Team Queue

Queues and *Priority Queues* are data structures which are known to most computer scientists. The *Team Queue*, however, is not so well known, though it occurs often in everyday life. At lunch time the queue in front of the Mensa is a team queue, for example.

In a team queue each element belongs to a team. If an element enters the queue, it first searches the queue from head to tail to check if some of its *teammates* (elements of the same team) are already in the queue. If yes, it enters the queue right behind them. If not, it enters the queue at the tail and becomes the new last element (bad luck). Dequeuing is done like in normal queues: elements are processed from head to tail in the order they appear in the team queue.

Your task is to write a program that simulates such a team queue.

Input

The input file will contain one or more test cases. Each test case begins with the number of teams t ($1 \leq t \leq 1000$). Then t team descriptions follow, each one consisting of the number of elements belonging to the team and the elements themselves. Elements are integers in the range 0 - 999999. A team may consist of up to 1000 elements.

Finally, a list of commands follows. There are three different kinds of commands:

- `ENQUEUE` x - enter element x into the team queue
- `DEQUEUE` - process the first element and remove it from the queue
- `STOP` - end of test case

The input will be terminated by a value of 0 for t.

Warning: A test case may contain up to 200000 (two hundred thousand) commands, so the implementation of the team queue should be efficient: both enqueing and dequeuing of an element should only take constant time.

Output

For each test case, first print a line saying "`Scenario #`k", where k is the number of the test case. Then, for each `DEQUEUE` command, print the element which is dequeued on a single line. Print a blank line after each test case, even after the last one.

Sample Input

```
2
3 101 102 103
3 201 202 203
ENQUEUE 101
ENQUEUE 201
ENQUEUE 102
ENQUEUE 202
ENQUEUE 103
ENQUEUE 203
```

```
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
STOP
2
5 259001 259002 259003 259004 259005
6 260001 260002 260003 260004 260005 260006
ENQUEUE 259001
ENQUEUE 260001
ENQUEUE 259002
ENQUEUE 259003
ENQUEUE 259004
ENQUEUE 259005
DEQUEUE
DEQUEUE
ENQUEUE 260002
ENQUEUE 260003
DEQUEUE
DEQUEUE
DEQUEUE
DEQUEUE
STOP
0
```

Sample Output

```
Scenario #1
101
102
103
201
202
203

Scenario #2
259001
259002
259003
259004
259005
260001
```