Problem 6: Team Queue

(Medium)

(Adapted from UVa 00540)

A queue is a data structure that is 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 Format

The input file will contain one or more test cases. Each test case begins with the number of teams t. Then t team descriptions follow, each one consisting of the number of elements belonging to the team, n_i for team number i, and the elements themselves.

Finally, a list of C commands follows, where C is to be inferred from the number of commands that follow. 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 team queue
- `STOP`: end of test case

Note that a `DEQUEUE` command will **never be called when the queue is empty**.

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

Constraints

- There are at most 128 test cases
- $1 \leq t \leq 1000$
- Each n_i satisfy $1 \leq n_i \leq 1000$
- Each element e is an integer that satisfies $0 \leq e \leq 999999 = 10^6 - 1$
- $2 \leq C \leq 200000$

The time limit for this problem is 3 seconds.

***Warning**: A test case may contain up to 200,000 commands, so the implementation of the team queue should be efficient: both enqueuing and dequeuing of an element should only take constant time.*

Output Format

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
```