Lab 02
Quacks

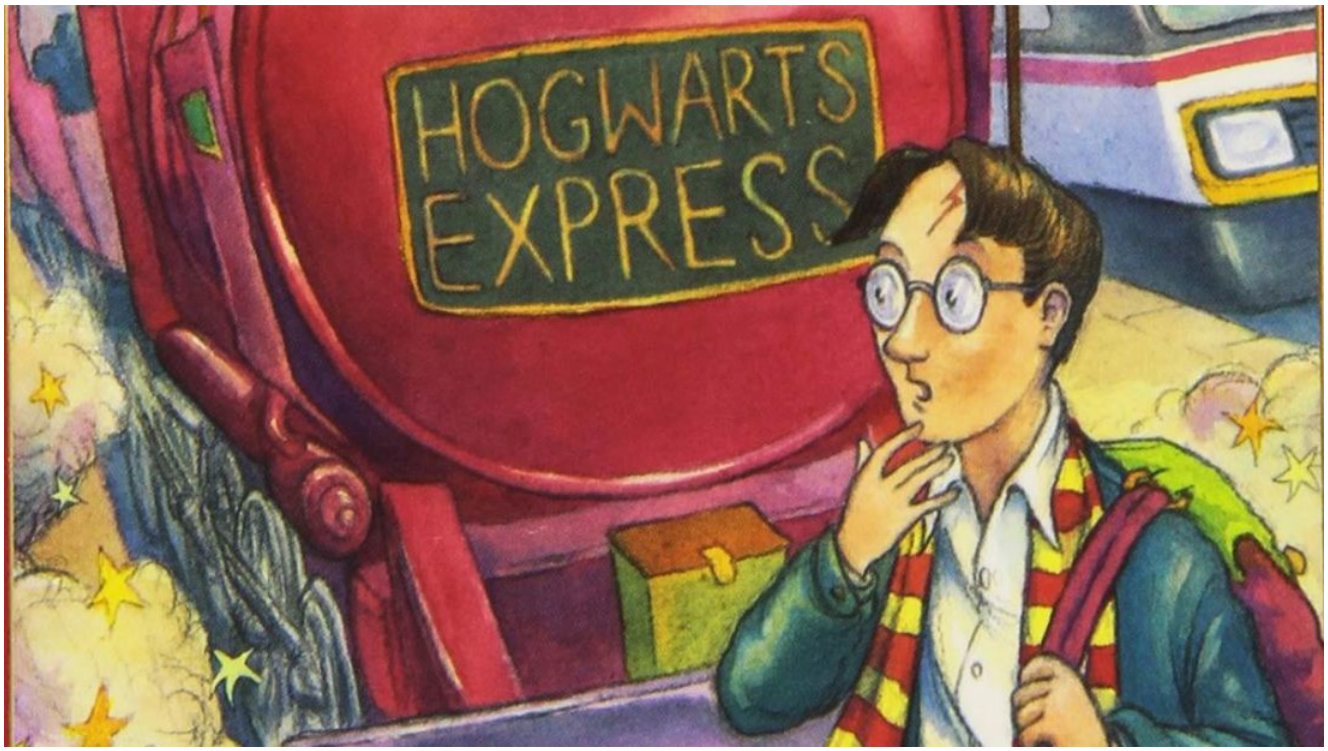
CSE 4304
DATA STRUCTURES LAB

JANUARY 22, 2020

The tasks of this lab will be solved and evaluated in Hackerrank!

1. If you do not have an account in Hackerrank, sign up
2. If you do have an account, log in
3. Go to Group A/Group B and sign up
4. Start solving!

The Philosopher's Stone



Harry and some other students are standing in the King's Cross station to board the train to Hogwarts. The train arrives. The students create a queue in front of the gate of the train. Each student is carrying a pet (either an owl, or a cat, or a toad). Since it's their first day, all of them are curious to see what the students in front of them have. Sadly, due to height differences, their view can be blocked.

The queue of the students can be considered as a straight horizontal line (from left to right). The leftmost student is standing in front and the rightmost student is standing in the back of the queue. So the students in the queue are trying to look past the students on their left. Student A will block the view of student B if the following conditions are satisfied:

- Student A is on the left of student B in the queue
- Student A is taller than student B

The student standing in front of the queue can only see his/her own pet. The student standing in the second, if s/he is taller than the first one, s/he can see two pets. However, if s/he isn't taller than the first one, s/he can see one. Same goes for all the students.

Now given the height of the students standing in the queue, you need to find the number of pets each student standing in the queue can see.

Input

First line contains an integer T, denoting the number of test cases.
For each test case, there will be an integer N, denoting the number of students. Following line will contain N space-separated integers denoting the height of the students.

Output

For each test case, print N space-separated integers denoting the number of pets each student can see.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N \leq 10^6$
- $1 \leq \text{height} \leq 10^8$

Sample

Sample Input	Sample Output
1 7 100 80 60 70 60 75 85	1 1 1 2 1 4 6

Hint

Check the following url.

The Chamber of Secrets

"Follow the spiders"

- Rubeus Hagrid

Following the advice of Hagrid, Harry and Ron head to the Forbidden Forest. There they meet Aragog, the acromantula, who tells them about Hagrid's innocence. But Aragog only loyal to Hagrid and considers all other humans as prey. So he asks Harry and Ron to fight a few rounds against the army of spiders.

Aragog lines up N spiders to attack them. He marks the spiders with their position. So the first spider is marked with 1, second one is marked with 2 and so on. Each spider has a strength value associated to it. For each round, Aragog chooses the first X spiders standing in front of the line. If there are less than X spiders the line, all of them are chosen.

Harry and Ron devised a strategy. They will attack the spider with the maximum strength, from the X chosen spiders, killing it in the process. If there are multiple spiders having the maximum strength, the one that comes first in the line, is selected and killed. This strikes fear among the remaining $(X-1)$ spiders. As a result, their strength is reduced by 1. However, if at any moment, strength of any spider becomes 0, it can't be decremented any further, it remains the same for the rest of the rounds. After each round, the remaining spiders go stand at the end of the line (in the same order they were before). The fight finishes after X rounds.

Harry and Ron know the strength of the N spiders standing in the queue. Now for each round, they want to know which spider to attack. They can read the marks made by Aragog. Your job is to help them find the spider to attack at each round.

Input

The first line consists of two space-separated integers N and X, denoting the number of spiders initially chosen and the number of spiders selected for each round, respectively.
The next line consists of N space-separated integers denoting the strength of each spider.

Output

For each round, output the mark of the spider with the highest strength. Here, mark refers to the index (1-based) at which the spider was present in the initial line.

Constraints

- $1 \leq X \leq 316$
- $X \leq N \leq X \times X$
- $1 \leq \text{Strength} \leq X$

Sample

Sample Input	Sample Output
6 5 1 2 2 3 4 5	5 6 4 1 2

Explanation

The initial line: 1, 2, 2, 3, 4, 5
In the 1st round, [1, 2, 2, 3, 4] are selected. The spider with strength 4 (Initial index 5) is killed. Rest are reduced by 1 and added to the line. So the line looks like: 5, 0, 1, 1, 2
In the 2nd round, [5, 0, 1, 1, 2] are selected. The spider with strength 5 (Initial index 6) is killed. Rest are reduced by 1 and added to the line. So the line becomes: 0, 0, 0, 1
In the 3rd round, [0, 0, 0, 1] are selected. The spider with strength 1 (Initial Index 4) is killed. Rest are reduced by 1 and added to the line. So the line looks like: 0, 0, 0
In the 4th round, [0, 0, 0] are selected. The spider with strength 0 (Initial Index 1) is killed. Rest are reduced by 1 and added to the line. So the line looks like: 0, 0
In the 5th round, [0, 0] are selected. The spider with strength 0 (Initial Index 2) is killed. Rest are reduced by 1 and added to the line. So the line looks like: 0.

Hint

Make a queue, whose each element will have a pair of values. First denoting the strength of spider, and second denoting their initial position in the queue. Now simply simulate all X iterations, and in each of those dequeue X elements, find the first one having maximum strength, print its original position, and enqueue the rest back in the queue with their strength decremented by 1, or 0 if their strength is already 0.

The Goblet of Fire

Hogwarts is hosting the legendary event know as the Triwizard Cup where four magical schools, Beauxbatons Academy of Magic, Durmstrang Institute, Hogwarts School of Witchcraft and Wizardry, and Incantation United Thermopolis are going to compete against each other facing some dangerous challenges. Champions from each school, selected by the Goblet can participate in this prestigious competition. To achieve the honor, students have formed a long queue to submit their names. Each student of the school have a different student IDs. Whenever a new student comes, s/he will search for their schoolmate from the end of the queue. As soon as s/he will find any of the schoolmate in the queue, s/he will stand behind them. Otherwise, s/he will stand at the end of the queue. At any moment, a student in front of the queue can come forward and put his/her name in the Goblet and leave. So there are two types of operations:

1. `E x y`: A new student of school x whose student ID is y will stand in the queue according to the method mentioned above.

2. `D`: A student standing in front of the queue comes to put his/her name in the Goblet and leaves.

Now the headmaster of Hogwarts, Professor Albus Dumbledore has asked you to keep a list of students to track which students have come and who will put his/her name in the Goblet next. Are you up to the task?
Note: The number of type-2 operations will never be greater than the number of type-1 operations at any point of time.

Input

First line contains an integer Q, denoting the number of operations.
The next Q lines contain one of the 2 types of operations.

Output

For each operation of type-2, print two space-separated integers, the front students school and student ID.

Constraints

- $1 \leq x \leq 4$
- $1 \leq y \leq 50000$

- $1 \leq Q \leq 10^5$

Sample

Sample Input	Sample Output
5 E 1 1 E 2 1 E 1 2 D D	1 1 1 2

Explanation

After the 1st operation, student of 1st school with student ID 1 will stand in the front of the queue as the queue is initially empty.
After the 2nd operation, student of 2nd school with student ID 1 will behind the first student as there is no other member of the same school in the queue.
After the 3rd operation, student of 1st school with student ID 2 will stand behind the member of his/her school in the queue in 2nd position (moving (2, 1) to 3rd position)
After the 4th operation, the student in the front (1, 1) will put his/her name in the Goblet and leave.
After the 5th operation, the student in the front (1, 2) will put his/her name in the Goblet and leave.

Hint

In this problem, the enqueue operation is defined as:
If a student of school x comes and there was already a student of school x in the queue, s/he will stand behind him/her in the queue, otherwise s/he will stand at the end of the queue.
Approach 1:
We simulate it the way it is given, that is, when a new student comes we search for his/her schoolmate in the current queue from the end of the queue. If we find a student from his/her school, we insert the new student after him/her and shift all other students one unit back in the queue. If no student of his/her school is there, we insert the new student in the end of the queue. This would take $O(N)$ time, as we are scanning the entire queue and then shifting the elements. This approach would take $O(Q \times N)$ time which will be too slow for the problem.
Approach 2:
Say we have 4 queues, one for each school. We also have one more queue which keeps track of the schools in the original queue.
Example: Let's enqueue these 4 students: (2,1), (2,2), (1,1), (1,2).

```
school1Queue = [1, 2]
school2Queue = [1, 2]
school3Queue = [ ]
school4Queue = [ ]
schoolSerialQueue = [2, 1]
```

`schoolSerialQueue` keeps track of which school is presently at what position in the original queue.
If we do two type-2 operations:

```
school1Queue = [1, 2]
school2Queue = [ ]
school3Queue = [ ]
school4Queue = [ ]
schoolSerialQueue = [1]
Say we now enqueue (2, 3).
```

```
school1Queue = [1, 2]
school2Queue = [3]
school3Queue = [ ]
school4Queue = [ ]
schoolSerialQueue = [1, 2]
```

When we do any enqueue operation, we enqueue the student in it's school's queue. If there was no student of his school till now in the original queue, we enqueue his school in the queue q. All this takes $O(1)$ time as we are only doing enqueue operation and no shifting is required.

When we do dequeue operation, we simply dequeue the student from his school's queue. If after this the school's queue becomes empty, we dequeue the school from `schoolSerialQueue`. This also takes $O(1)$ time.