CIMS Printer

Due to some reason there is only one printer still working at Courant Institute of Mathematical Sciences. There are tons of printing jobs queuing for that printer and you may have to wait for a very very long time to get your paper printed.

Printing jobs are assigned a priority between 1 and 9 (inclusive) where 9 is highest priority and 1 is the lowest. The printing system manages the jobs as follows.

- Select the first job from the queue.
- If there is some job in the queue with higher priority than the selected job then move the selected job to the end of the queue without printing it.
- Otherwise, print that job.

Now your problem is that it becomes so tricky to determine when your paper will be printed. You are to write a program to figure it out. The program is given the current printing queue and the position of you job in the queue. It should output how long it will take until your paper is printed. Assume that there are no additional jobs added to the queue and printing a single job takes exactly one minute (regardless of its size) and that the queue operations happen instantly (i.e., there is no additional time spent on them).

Input
The first line of input contains two integers `N` and `M` where `N` stands for the number of jobs in the print queue (`1 <= N <= 100`) and `M` is the position of your job `(0 <= M <= N - 1` ; with 0 being the first position in the queue). The second line has `N` integers in the range, giving the priorities of jobs in the queue from the first position to the last.

Output
You should print **one line** with a single number, the number of minutes until your job is printed.

Example 1

```
Input:
1 0
5

Output:
1
```

Example 2

```
Input:
4 2
1 2 3 4

Output:
2
```

Example 3

```
Input:
6 0
1 1 9 1 1 1

Output:
5
```

Ferry

Ferries used to carry cars across the river. In your village, there is still a ferry that can take up to `N` cars and needs `T` minutes to cross the river. A car may arrive at either river bank and wait to be carried to the opposite bank. The ferry operates continuously between the banks as long it is carrying at least one car or there is at least one car waiting on either side. Whenever the ferry arrives at one bank, it unloads cars carried and loads up to `N` cars waiting at that bank. When there are more than `N` cars waiting, they are loaded on the first-come-first-serve basis. If there is no car waiting on either bank, the ferry stops and waits until one car arrives. The ferry is initially on the left bank. You are asked to answer at what time does each car arrive at the other bank.

Input
The first line of input contains three integers `N`, `T` and `M` (`1 <= N, T, M <= 10,000`). Each of the following `M` lines gives the arrival time of a car and the bank at which the car arrives (`left` or `right`). The cars are ordered by their arrival times (so the arrival times are non-decreasing) and the time spent on loading and unloading can be ignored.

Output
For each car, you should print one line containing one number, the time at which the car is unloaded at the opposite bank.

Example 1

```
Input:
2 10 10
0 left
10 left
20 left
30 left
40 left
50 left
60 left
70 left
80 left
90 left

Output:
10
30
30
50
50
70
70
90
90
110
```

Example 2

```
Input:
2 10 3
10 right
25 left
40 left

Output:
30
40
60
```

Line Up

Queuing, it's what the British are renowned for doing - and doing very well. Better than anyone else in the world, if reputation is to be believed. Today, queues can be found just about anywhere (even in the virtual world).

Standard *Queues* are well known to most computer science students. It's a first-in-first-out data structure. Kobayashi is already familiar with it. But she is assigned to write some code to maintain a variant of Standard Queues: **Group Queues**.

In a group queue each element belongs to a group. If an element is pushed to the queue and there is no element that belongs to the same group in the queue, it will be placed behind the last element in the queue. Otherwise, if there are already some elements that belong to the same group, it should be placed immediately behind them.

Dequeuing works the same as int the standard queue: We will remove the first element according to the current order from head to tail in the queue.

Kanna is preparing for her school's Sports Festival, and she wants Kobayashi to come to this important event. But Kobayashi will not be able to attend unless she can finish her program early. As Kobayashi's colleague and close friend, you decide to help her. You need to write a program that simulates such a group queue.

Input

The 1st line contains an integer `n` (`1 <= n <= 1,000`) equal to the number of groups, as described above.

Then `n` group descriptions follow. Each group is described on one line by an integer `k` (the number of elements belonging to the group) followed by `k` integers (the indexes of the elements in this group).

Elements are indexed with integer in the range `[ 0 , 999,999 ]`. And a group may consist of up to `1,000` elements.

Finally, a list of commands follows. There are 3 different kinds of commands:

- `Push ind` : Push the element with index `ind` into the group queue.
- `Pop` : Output the first element and remove it from the queue.
- `Shutdown` : Stop your program (end of input).

There may be up to `200,000` commands in the input file, so the implementation of the group queue should be efficient: both enqueuing and dequeuing of an element should only take a constant time.

Output

For each `Pop` command, print an integer, the index of the element which is dequeued, followed by a newline.

Example 1

```
Input:
2
3 1 2 3
3 4 5 6
Push 1
Push 4
Push 2
Push 5
Push 3
Push 6
Pop
Pop
Pop
```

```
Pop
Pop
Pop
Shutdown

Output:
1
2
3
4
5
6
```

Example 2

```
Input:
2
3 1 2 3
3 4 5 6
Push 1
Push 2
Push 4
Pop
Pop
Push 3
Push 5
Pop
Pop
Push 6
Pop
Pop
Shutdown

Output:
1
2
4
5
3
6
```

Labor Strike

There are `n` labor unions in New York City. You have learned that the labor union `i` will undertake a strike every `s_i` days.

For example, assume the 1st day is Sunday, and labor union 1 undertakes a strike ever 4 days. Then there will be a strike on the 4th day (the Wednesday), the 8th day (following Sunday) and so forth. (See the table below.)

Because the MTA employees support all the unions, the subways do not work if at least one of the unions is on strike. The only exception to this are Fridays and Saturdays: even if a union has a planned strike for that day, the subways will still operate as usual.

You need to figure out the number of days that you can not take the subway in the following `m` days.

You can assume the 1st day is always Sunday.

Days	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
	Su	Mo	Tu	We	Th	Fr	Sa	Su	Mo	Tu	We	Th	Fr	Sa
Union 1				x				x				x		
Union 2			x			x			x			x		
Union 3								x						
No subway			x	x				x	x			x		

Input

The 1st line contains an integer `m` (`7 <= m <= 3650`) equal to the number of days for which you wish the make the calculation.

The next line contains another integer `n` (`1 <= n <= 100`) representing the number of labor unions.

The *i*-th of the next `n` lines contain a positive integer `s_i` (which will never be a multiple of 7) giving the striking period of labor union *i*.

Output

Output one integer followed by a newline: the total number of days with strikes.

Example 1 (shown in the table above)

```
Input:
14
3
4
3
8

Output:
5
```

You can find there will be strikes on the 3rd, 4th, 6th, 8th, 9th, and 12th days. So the subway will not operate on the 3rd, 4th, 8th, 9th, and 12th days (since the 6th day is a Friday).

Example 2

```
Input:
100
4
12
15
25
40

Output:
15
```

Ruthless War

Light Kingdom is fighting a ruthless war against Conflict Empire. As a general of Light Kingdom, you decided to attack the enemy with a linear formation of soldiers. You ordered that each soldier in the attack line should protect their two nearest neighbors in the line. These two neighbors are called pals. Note that the leftmost soldier does not have a left pal and the rightmost soldier does not have a right pal. If the either of the pals of a soldier is killed, then the next living neighbor to the left/right becomes that soldier's new pal.

As the time passes, the battle becomes much more fierce and many soldiers are being killed by light sabers, blaster pistols and magical spells. In order to keep soldier aware of their pals you need to update them about their new pals after receiving each loss report.

Input
The first line of input contains two integers `N` and `M` (`1 <= N <= M <= 100,000`) representing the number of soldiers in the attack line and the number of loss reports respectively. Soldiers are numbered from `1` to `N` according to their positions in the attack line where `1` identifies the leftmost soldier and `N` identifies the rightmost soldier. Each of the following `M` lines describes a loss report and contains two integers `L` (left) and `R` (right), meaning that soldiers from `L` to `R` were killed (`1 <= L <= R <= N`). It is guaranteed that those soldiers were alive until that moment.

Output
You should print `M` lines. In the *i*-th line should contain the new pal relationships formed after removing from the attack line the soldiers who were killed according to the *i*-th loss report. That is, for the loss report `L R` , print the first living soldier to the left of `L` and the first living soldier to the right of `R` . If there is no surviving soldier in some direction, print the character `*` instead.

Example 1

```
Input:
1 1
1 1

Output:
* *
```

Example 2

```
Input:
10 4
2 5
6 9
1 1
10 10

Output:
1 6
1 10
* 10
* *
```

Example 3

```
Input:
5 1
1 1

Output:
```

* 2