Problem Q: Queue skipping

Recently there was a rumor that on Monday the meat store will actually have some meat. It's Monday, half past one in the morning. The entire town is already waiting for the store to open. More precisely, there are n people waiting in a line. The people are numbered 1 through n in line order, with person 1 being the one currently closest to the door of the store.

During the next few hours, at some moments one of the people will use some trick in order to skip to the beginning of the queue. Examples of such tricks include:

- Look over there, isn't that *[famous TV show host]*?
- Please let me through, I'm with a small child!
- I stood here before, I just went to the bathroom.
- Hey, look at that guy trying to skip the queue, I'll go and throw him out!

Problem specification

You are given the sequence of people skipping to the front of the queue. After all those events, the meat store will open. People will enter the store one at a time. Inside, they will learn that the meat didn't arrive (again!), so they will go home empty-handed.

Find out who will waste the most time waiting for nothing. In other words, find out who will be the last person in the queue after all the skipping to the front is over.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with a line containing two positive integers n and e: the number of people and the number of events. The rest of the test case consists of e lines, each containing a single positive integer: the number of a person who just moved to the front of the queue.

In the **easy subproblem Q1** you may assume that $t = 100$ and $1 \leq n, e \leq 100$.

In the **hard subproblem Q2** you may assume that $t = 100$ and $1 \leq n, e \leq 10^5$.

Note that you cannot download the input for subproblem Q2 directly. Instead, we have provided a small Python 2 program that will generate the file `q2.in` when executed.

Note

In the real contest, some large input files may be provided in the same way as the input `q2.in` in this practice problem. Please make sure you are able to generate it.

Output specification

For each test case output a single line with a single positive integer: the number of the last person in the queue when the meat store opened.

Example

input

```
2

3 4
3
1
3
2

7 1
1
```

output

```
1
7
```

In the first example test case, the queue changed as follows:

- *at the beginning: (1,2,3)*
- *person 3 skips to the front: (3,1,2)*
- *person 1 skips to the front: (1,3,2)*
- *person 3 skips to the front again: (3,1,2)*
- *person 2 skips to the front: (2,3,1)*
- *thus, the last person at the end is person 1.*

In the second example test case, person 1 skipping to the front of the queue doesn't change the queue at all. In particular, the last person in the queue is still person 7.

Problem R: Russian roulette

You went out to celebrate with your $n-1$ friends. You all got drunk and decided that playing Russian Roulette with a paintball gun sounds like a great idea.

Problem specification

The gun is a revolver with c chambers ($c \geq n$). Exactly $n-1$ of those chambers now contain a paintball, all others are empty.

The game has one other parameter: a positive integer constant k.

At the beginning of the game all n players stand around a circle. We will number the players 0 through $n-1$ as they're standing in clockwise order. Player number 0 takes the gun. The game is then played by repeating the following three steps until the gun becomes empty:

1. The person who has the gun uses it on themselves. This means that they spin the cylinder to choose one of the chambers uniformly at random and then they point the gun on themselves and pull the trigger.
2. If the chosen chamber was empty, there are exactly k rounds of passing the gun. In each round the person who has the gun passes it clockwise to the next person who is still playing the game.
3. If the chamber contained a paintball, the person is hit by the paintball and loses the game. In this case, they just pass the gun to the next playing person clockwise and they leave the game.

The winner is, obviously, the only person who doesn't get shot.

You are given the parameters n, c, and k. Compute where you should stand at the beginning of the game in order to maximize your probability of winning. Also, compute the exact value of that probability.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each case consists of a single line with the integers n, c, and k. You may assume that $1 \leq k \leq n \leq c$.

In the **easy subproblem R1** you may assume that $c \leq 10$.

In the **hard subproblem R2** you may assume that $c \leq 1000$.

Output specification, easy subproblem R1

For each test case, print one line containing two space-separated numbers: your position at the beginning of the game which maximises the probability of winning and the probability of winning when standing at that position.

Output the probability as a floating-point number with at least five decimal places. Answers with absolute or relative error up to 10^{-4} will be accepted.

You may assume that in each test case the optimal position is unique and that the probability of winning for the optimal position differs from the probability for any other position by more than 10^{-6}.

Output specification, hard subproblem R2

For each test case, print one line containing two space-separated numbers: your position at the beginning of the game which maximises the probability of winning and **a number that encodes** the probability of winning when standing at that position, as defined below.

If there are multiple optimal positions, output the **smallest** one of them.

Let the maximum probability of winning be an irreducible fraction of the form p/q. In order to output this probability, print the number pq^{-1} modulo $(10^9 + 9)$, where q^{-1} is the modular inverse of q modulo $10^9 + 9$. It is guaranteed that the maximum probability of winning can be expressed as a fraction and that the modular inverse of q exists.

Example, easy subproblem R1

input

```
3

3 3 1

2 2 2

4 5 3
```

output

```
2 0.5076923077
1 1.0000000000
2 0.4105090312
```

In the first example, each player passes the gun to the next person clockwise. Intuitively, you should go as late as possible – stand counter-clockwise from the person who starts. This turns out to be the optimal strategy. The exact probability of winning for player 2 *is* $33/65$.

In the second example, player number 0 *keeps on using the gun until they lose the game. (After each unsuccessful shot the gun gets passed twice, which brings it back to player* 0*.) Hence, the probability of winning is* 1 *for player* 1 *and* 0 *for player* 0.

In the third example, the following happens, in order:

- *While there are four players, each time they pass the gun clockwise three times, it is as if they passed it counter-clockwise once. The gun essentially travels counter-clockwise around the circle until a person shoots themselves.*
- *The next person clockwise gets the gun. This person now keeps trying to shoot themselves until they succeed – which they eventually will.*
- *The remaining two people take alternating shots until one of them loses.*

The probabilities of winning for players 0, 1, 2, 3 *are* $40/609$, $100/609$, $250/609$, *and* $219/609$, *respectively.*

Example, hard subproblem R2

For the sample input shown above the correct output for subproblem R2 looks as follows:

```
2 661538468
1 1
2 348111662
```

In the first test case the exact answer is $33/65$. *Since* $65^{-1} \equiv 323\,076\,926 \bmod (10^9 + 9)$, *the second number in the first line of the sample output is* $(33 \cdot 323\,076\,926) \bmod (10^9 + 9) = 661\,538\,468$.

Problem S: St. Ives

As I was approaching St. Ives,
Out came a man with seven wives,
Each wife had seven sacks,
Each sack had seven cats,
Each cat had seven kits:
Kits, cats, sacks, and wives,
How many were there going to St. Ives?

Problem specification

Above you see one of many versions of an old nursery rhyme. The **easy subproblem S1** is about a generalized version of the riddle contained in the nursery rhyme.

The nursery rhyme contains 5 types of objects: the man, the wives, the sacks, the cats, and the kits (i.e., kittens). In the generalized version there are n types of such objects. We will number these types 0 through $n-1$, in order.

In the nursery rhyme there is one man, and each object other than a kitten has 7 objects of the following type. In the generalized version there is one object of type 0, and each object of type i has a_i objects of type $i+1$.

In the **easy subproblem S1** you are given the number n and the numbers $a_0, \dots, a_{n-2}$. Compute and output the answer to the riddle.

In the **hard subproblem S2** the input data is the same but the question you have to answer is different. The question will be revealed to you after you solve the subproblem S1. (More precisely, you will find the question for S2 in the evaluation details of an accepted submission to S1.)

Input specification

The first line of the input file contains an integer $t = 100$ specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of two lines. The first line contains the positive integer n. The second line contains the nonnegative integers $a_0, \dots, a_{n-2}$. You may assume that $1 \le n \le 10$ and that for each i, $0 \le a_i \le 10$. Note that for $n = 1$ the second line of a test case will be empty.

Output specification

For each test case output a single line with a single integer: the total number of objects that were, according to the riddle, going to St. Ives.

Example

input

```
1

4
0 10 2
```

output

```
1
```

In the example test case there is a man with zero wives, each wife has 10 sacks, and each sack has two cats.