Problem A - Attack and Split

There are slimes live in a peace world. Sometimes they split: a slime with size x can split into two slimes of positive integral sizes y and z with $x = y + z$. Sometimes they attack each other, two slimes of sizes p and q will become $r = p \oplus q$, where $\oplus$ denotes an xor operation.

Given sizes of the slimes in this world, is it possible for them to disappear after several attacks and/or splits?

Input

The first line contains an integer T $(1 \le T \le 100)$, indicating the number of test cases.

For each test case, the first line contains an integer n $(1 \le n \le 100)$, indicating the number of slimes, followed by n positive integers $a_1, a_2, \cdots, a_n$ denoting the sizes of slimes. $(1 \le a_i \le 10^9)$.

Output

For each test case, output `Yes` or `No` for answering the question.

Sample Input

```
3
1 1
2 9 17
3 12 15 19
```

Sample Output

```
No
Yes
Yes
```

Useless Note

When you split a slime with some upper bounds, you'll get a sublime.

If you reverse a slime toward some limit, it will smile.

Problem B - Dreamoon's Criterion

Dreamoon loves to solve problems. He even sets a simple criterion to evaluate the difficulty of problems: If he spent more than t minutes to solve a problem, then this problem is considered *hard*. Otherwise, the problem is considered *easy*.

Little Tomato had set up n problems, and asked Dreamoon for help to test whether these problems are easy or hard. He picked out some problems among them, and asked Dreamoon to solve them one after another. However, he noticed that Dreamoon would get tired after solving each problem.

Let's define a *tiredness value* v, and a *tiredness gap* k. Before Dreamoon solved the first problem, the tiredness value $v = 0$. Whenever he solved one problem, the tiredness value is increased by a tiredness gap k immediately. That is, $v \leftarrow v + k$ after each problem is solved.

Now Little Tomato has estimated the time $a_1, a_2, \cdots, a_n$ that Dreamoon would need to solve for each problem, assuming it is the first problem for Dreamoon to solve. Then Dreamoon would need $a_i + v$ minutes to solve for problem i with tiredness value v.

For example, if there are five problems with estimated solving time $7, 3, 6, 8, 5$, and tiredness gap $k = 2$, difficulty threshold $t = 9$. Then by asking Dreamoon to solve the third, second and fourth problem, he will finish in $6, 5, 12$ minutes respectively, and those problem will be considered as easy, easy and hard respectively.

Little Tomato wondered at most how many problems that Dreamoon will consider as easy, under the different settings of difficulty threshold t?

Input

The first line contains an integer T ($1 \leq T \leq 100$), indicating the number of test cases.

For each test case, the first line contains two integers n, k ($1 \leq n \leq 100000; 1 \leq k \leq 10^9$), followed by n positive integers $a_1, a_2, \cdots, a_n$ ($1 \leq a_i \leq 10^9$) on the next line.

The third line contains an integer Q ($1 \leq Q \leq 100000$), denoting the number of queries. For each of the next Q lines, there is an integer t ($1 \leq t \leq 10^9$).

The input file will be at most 10MB.

Output

For each query, please output the maximum number of problems that can be declared easy by Dreamoon in a line.

Sample Input

```
1
5 2
7 3 6 8 5
2
9
1
```

Sample Output

```
4
0
```

Problem C - Heap Like Sort

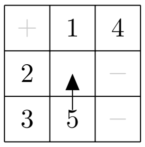
Little Heap likes to sort things. He likes to sort some sort of things using heaps. One day, Little Heap gets a large square board. Moreover, there are several distinct numbers on the anti-diagonal cells.

An **open cell** is an empty cell that there are no filled cell to its left or above or any north-west direction. A **closed cell** is an empty cell that there are no filled cell to its right or below or any south-east direction. An **inner cell** is any cell on the board that is neither an open cell nor a closed cell. For example, open cells below is marked by $+$, and closed cells are marked by $-$. It is possible for a cell that is recognized as open and closed at the same time. We call a cell a **open-only cell** if it is an open cell but not a closed cell.

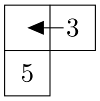
+	+	+	+	+	4
+	+	+	+	1	−
+	+	+	3	−	−
+	+	5	−	−	−
+	6	−	−	−	−
2	−	−	−	−	−

Little Heap start moving these numbers by the following algorithm:

1. Whenever there is an **inner empty cell** with its right cell and bottom cell filled with some integers (or a closed cell). Little Heap pick the smaller one among the two (or the only one) and drag them into this cell.



2. If there is no such empty cell described above, pick any one **open-only cell** that has *no* other open-only cells to its right or below. In this case there must be at least one cell to its right or below is filled. If after dragging the number from that filled cell, all numbers in the same row or same column remains increasing (from left to right, or from up to down), Little Heap can drag the number from the filled cell into it.



3. If there is no empty cell described in 1. or 2., the whole process stops.

Little Heap finds that if originally all the numbers are on anti-diagonal cells, then after continuously applying the first two rules, the numbers in the same row or the same column remain increasing (from left to right, or from up to down).

It is known that, no matter how you choose these empty cells in order, the outcome will always be the same.

Now Little Heap had performed some steps according to the algorithm, please continue his algorithm and output the number of non-empty cells in each rows after it stops.

Input

The first line contains an integer T $(1 \le T \le 100)$, indicating the number of test cases.

For each test case, the first line contains two integers n, m $(1 \le n, m \le 100)$ indicating that we are showing the first n rows and the first m columns of the board.

Each of the next n lines have m integers separated by whitespaces. The j-th number in the i-th row a_{ij} $(0 \le a_{ij} \le nm)$ denote the number inside that cell. If $a_{ij} = 0$, it means that the cell is empty. All positive values will be distinct, and the input will be valid.

Output

For each test case, output K integers, one at a line: the number of non-empty cells in each non-empty row after the algorithm stops.

Sample Input

```
3
3 3
0 0 1
0 2 0
3 0 0
6 6
0 0 0 0 0 4
0 0 0 0 1 0
0 0 0 3 0 0
0 0 5 0 0 0
0 6 0 0 0 0
2 0 0 0 0 0
3 5
```

```
1 2 3 4 5
6 7 0 0 0
8 0 0 0 0
```

Sample Output

```
1
1
1
3
1
1
1
5
2
1
```

Problem D - A Parking Problem

There are n parking lots along a one-way road, numbered from 1 to n. Parking lot number 1 is located at the entrance of this road, and parking lot number n is located near the exit of the road. There are m drivers wish to park their cars in to these parking lots. Every driver has a preferred parking lot. Among these m drivers, there are exactly a_1 of them preferred the first parking lot, a_2 of them preferred the second parking lot, $\cdots$, and so on.

Drivers (with car, of course) may appear at the entrance of the road in some order. Each driver directly drives his/her car to their preferred parking lot immediately after he/she enter the road. If the parking lot is vacant, the driver will park his/her car happily. If the parking lot is occupied, the driver will seek the first available parking lot along the road, and park it. In the worst scenario, if a driver arrives at the end of the road, he/she leaves immediately.

Given preferences of all drivers, now we want to give out parking permits to n drivers among them, so that if these drivers arrive the parking lots **in any order**, all of them can always park their cars.

How many possible ways we can give out the parking permits?

Input

The first line of the input contains an integer T $(1 \leq T \leq 100)$, indicating the number of test cases.

For each test cases, first line contains an integer n $(1 \leq n \leq 50)$. Next line contains n integers $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 50)$, where a_i denoting that there are exactly a_i number of drivers having parking preference i.

Output

For each test case, output the number of possible subsets of size n among the drivers so that they can always park their car modulo 1000000007.

Sample Input

```
1
3
2 1 2
```

Sample Output

```
7
```

Problem E - Kirino in Google Code Jam

Kirino is a cute junior school girl. She has a brother who rents a house and lives alone for preparing university entrance examination. Kirino likes her brother very much, so she will bring breakfast to her brother every day.

Kirino promised her brother that she will bring breakfast to him at 10:00 a.m. this Saturday. But after making this promise, she found Google Code Jam Round 1A is hosted from 9:00 to 12:00 that day! Thus she must solve all problems in only forty minutes for qualifying to next round (she need another twenty minutes to prepare breakfast and go to the place of her brother).

Then Saturday came. Kirino felt lucky because all problems were easy to her. She completed all three problems in 30 minutes. But Kirino found that there exists a testcase which makes her code for problem C fail and that she cannot resolve it in short time. Kirino thinks her brother is more important than Google Code Jam, so she submitted her wrong code and started to prepare breakfast for her brother.

When the result of code jam is released, Kirino is shocked that she passed all problems! She thought this was the bless from her love for her brother.

Now, the problem is coming. Can you generate a testcase such that Kirino's code will fail?

You can find the statement of the Google Code Jam problem here:

http://code.google.com/codejam/contest/4224486/dashboard#s=p2

And Kirino's code here:

https://code.google.com/codejam/contest/4224486/scoreboard#vf=1

Input

This problem has no input file.

Output

Please output a valid input with $T = 1$ for the problem (Google Code Jam 2015 Round 1A Problem C) which makes Kirino's code fail..

Sample Output

Following is a valid output for Google Code Jam 2015 Round 1A problem C. But it cannot challenge Kirino's code successfully.

```
1
5
0 0
10 0
10 10
0 10
5 5
```

Problem F - No challenge No life

Stephnie Dola is assistant of Sora. She is a hard-working girl, but she is usually bullied by Sora because of her stupid opinion on work. When she feel sad, she will find solutions of UVa problems written by other people from the Internet and challenge these solutions. She thinks the action can make her fell more comfortable.

Today she found following blog:

http://www.cnblogs.com/yours1103/p/3426212.html

She thinks the code of UVa12647 in this blog is wrong and she challenged it succeffully. Cound you do the same thing as Stephnie Dola do?

Note that you can find the problem statement of UVa12647 from here:

http://uva.onlinejudge.org/external/126/p12647.pdf

Input

This problem has no input file.

Output

Please output a valid input containing only one test case with $N \leq 1000$ for UVa12647.

Sample Output

Following is a valid output for UVa12647. But it cannot challenge the code from above blog successfully.

```
4 4
0 1 3 3
1 5 6 5
5 3 2 4
7 4 10 2
2
5
8
6
```

After Story

Stephnie Dola wanted to prove she not only can challenge code of others but also write a correct solution. So she also wrote a solution as the link: https://gist.github.com/dreamoon4/11bc833e01937bcb3088 . But her code still was challenged by Sora . . .

Problem G - Strange Permutations

Little Tomato loves permutations. One day he found an amazing property to the permutation $(1, 2, 3, \cdots, n)$: the neighboring two numbers are coprime to each other! (Of course k and $k+1$ having their greatest common divisor equal to 1.)

Now, given a partial permutation of $1, 2, \cdots, n$, with blanked positions denoted by 0, can you help Little Tomato to find the number of ways to fill in this part of permutation and such that no two neighboring integers having greatest common divisor greater than 1?

Input

The first line contains an integer T $(1 \leq T \leq 20)$ indicating the number of test cases.

For each test case, the first line contains an integer n $(1 \leq n \leq 20)$. The second line contains n integers which is guaranteed to be a part of some permutation of 1 to n.

Output

For each test case please output the answer modulo 1000000007.

Sample Input

```
1
8
1 2 0 0 0 0 7 8
```

Sample Output

```
1
```

Problem H - Tomato's Criterion

(The problem description is exactly the same as problem B except the last paragraph.)

Dreamoon loves to solve problems. He even sets a simple criterion to evaluate the difficulty of problems: If he spent more than t minutes to solve a problem, then this problem is considered *hard*. Otherwise, the problem is considered *easy*.

Little Tomato had set up n problems, and asked Dreamoon for help to test whether these problems are easy or hard. He picked out some problems among them, and asked Dreamoon to solve them one after another. However, he noticed that Dreamoon would get tired after solving each problem.

Let's define a *tiredness value* v, and a *tiredness gap* k. Before Dreamoon solved the first problem, the tiredness value $v = 0$. Whenever he solved one problem, the tiredness value is increased by a tiredness gap k immediately. That is, $v \leftarrow v + k$ after each problem is solved.

Now Little Tomato has estimated the time $a_1, a_2, \cdots, a_n$ that Dreamoon would need to solve for each problem, assuming it is the first problem for Dreamoon to solve. Then Dreamoon would need $a_i + v$ minutes to solve for problem i with tiredness value v.

For example, if there are five problems with estimated solving time $7, 3, 6, 8, 5$, and tiredness gap $k = 2$, difficulty threshold $t = 9$. Then by asking Dreamoon to solve the third, second and fourth problem, he will finish in $6, 5, 12$ minutes respectively, and those problem will be considered as easy, easy and hard respectively.

Little Tomato wondered at most how many problems that Dreamoon will consider as easy, under the different settings of tiredness gap k?

Input

The first line contains an integer T $(1 \leq T \leq 100)$, indicating the number of test cases.

For each test case, the first line contains two integers n, t $(1 \leq n \leq 100000; 1 \leq t \leq 10^9)$, followed by n positive integers $a_1, a_2, \cdots, a_n$ $(1 \leq a_i \leq 10^9)$ on the next line.

The third line contains an integer Q $(1 \leq Q \leq 100000)$, denoting the number of queries. For each of the next Q lines, there is an integer k $(1 \leq k \leq 10^9)$.

The input file will be at most 10MB.

Output

For each query, please output the maximum number of problems that can be declared easy by Dreamoon in a line.

Sample Input

```
1
5 9
7 3 6 8 5
2
2
3
```

Sample Output

```
4
3
```

Problem I - Cow Families

Farmer Jiang has m families of cows, and for the i-th family, there are a_i cows. In a lovely late summer weekend FJ arranged all $n = a_1 + a_2 + \cdots + a_m$ cows into exactly n pastures so that each cow could enjoy her own pasture!

After that, FJ marked each pasture by the cow's family number (which are conveniently numbered from $1, 2, \cdots, m$) and wrote down the sequence on paper. Surprisingly, he found that the **longest increasing subsequence** has exactly m numbers! Now FJ wonders, how many possible other arrangements of his cows there are such that this property would hold.

Help FJ calculate this number modulo 1000000007. Two arrangements are considered distinct if there is a pasture with a cow from a different family in each arrangement.

Input

The first line of input contains an integer T ($1 \leq T \leq 100$), indicating the number of test cases.

For each test case, the first line contains m ($1 \leq m \leq 50$).

The next line contains m integers separated by a whitespace. The i-th integer describes a_i ($1 \leq a_i \leq 50$), the number of cows in the i-th family.

Output

A single integer for each test case, the number of arrangements modulo 1000000007.

Sample Input

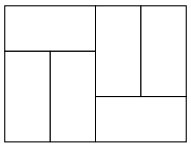
```
1
3
2 1 2
```

Sample Output

```
11
```

Problem J - Domino Tableaux

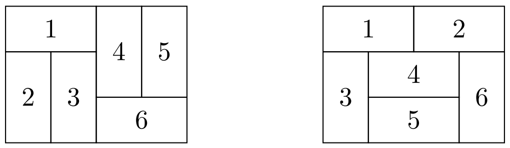
Remember the problem of domino tiling? Given an $m \times n$ rectangular board, we want to tile it with 1×2 dominos. For example, we can tile a 3×4 board with the following tiling:



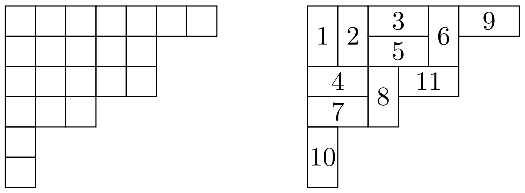
We want to give each of the dominos a unique sequential number, starting from 1, subjecting to the following condition:

- For two dominos X and Y, if there is any interior point of X, located at upper-left position to some interior point of Y, then the number of X must be strictly less than number of Y.

For example, the following are two valid domino tiling with numbers on a 3×4 board:



Let's consider a partial board like this:



The i-th row has exactly λ_i number of cells, and they satisfy

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq 0$$

We call the sequence $\lambda = (\lambda_1, \lambda_2, \cdots)$ a *partition*, and the *size* of the diagram is denoting by $n = |\lambda| = \lambda_1 + \lambda_2 + \cdots$.

Now Little Tomato gets a partition λ with $|\lambda| = n$, can you help him to count how many domino tilings with numbers are there in λ?

Input

The first line contains an integer T $(1 \leq T \leq 200)$ indicating the number of test cases.

For each test case the first line contains an integer m $(1 \leq m \leq 100000)$. The second line contains m non-increasing positive integers $\lambda_1, \lambda_2, \cdots, \lambda_m$. Their sum is guaranteed to be no more than 100000.

Output

For each test case, please output the answer modulo 1000000007.

Sample Input

```
3
1
9
4
3 3 3 3
6
7 5 5 3 1 1
```

Sample Output

```
0
30
73920
```

Problem K - Increasing Subsequences

Given a permutation of 1 to n, what is the length of the longest increasing subsequence? This problem is quite easy nowadays.

Given a permutation π of 1 to n, if you are allowed to pick k disjoint increasing subsequences, what is the maximum number of numbers you can pick? Suppose the permutation π can be separated into at least L disjoint increasing subsequences, please answer the previous question for every $k = 1, 2, \cdots, L$.

Note

It is easy to show that L will be equal to the length of the longest decreasing subsequence of π.

Input

The first line contains an integer T ($1 \leq T \leq 100$), indicating the number of test cases.

For each test case, first line contains an integer n ($1 \leq n \leq 30000$). Then the following n integers $\pi_1, \pi_2, \cdots, \pi_n$ denote a permutation of 1 to n.

Output

For each test case, output L lines, where L is described in the description. For each $k = 1, 2, \cdots, L$ output the desired answer in a line.

Sample Input

```
2
6
2 6 5 3 1 4
6
4 3 5 1 6 2
```

Sample Output

```
3
4
5
6
```

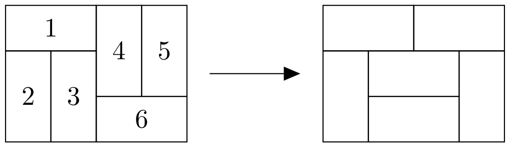
```
3
5
6
```

Problem L - Domino Rotations

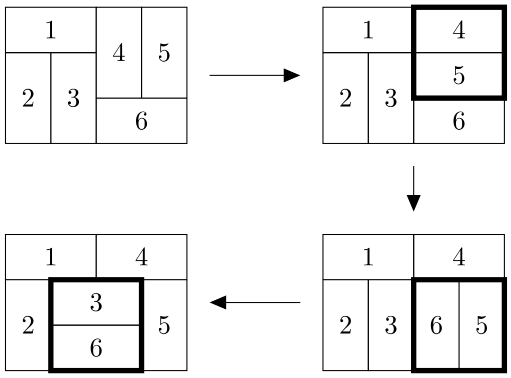
Consider two domino tiling on a $m \times n$ rectangular board. Is it possible to go from one to another by subsequently rotate some 2×2 squares formed by two vertical or horizontal dominos?

The numbers on the dominoes do not matter. They are only to help us clarify how the rotations work.

For example, if we are given the following two tilings:



Then we can achieve this task in three steps:



Now you are given two layouts of domino tilings, with label of numbers in the first one. Please find out the **minimum number** of rotations starting from the first one to the second one if possible.

Input

The first line contains an integer T ($1 \leq T \leq 100$), denoting the number of test cases.

For each test case, the first line contains two integers m, n ($1 \leq m, n \leq 90$) indicating the number of rows and the number of columns of the board. Then exactly $4m + 3$ lines follow. It consists of two domino tilings. Please refer to the sample input. All numbers will be distinct and will be represented in base-16.

Output

For each test case, please output a number r first, indicating the minimum number of rotations you can achieve. In case there is no solution, please output $r = -1$.

If there is a solution, please output another r lines. Each of these lines contains two base-16 integers x, y indicating the two dominoes that are to be rotated. The rotations will always be **right turns**. Any valid minimum length script of rotations will be accepted. Do not output leading zeroes.

Sample Input

```
1
3 4
+-+-+-+-+
|001|0|1|
+-+-+9+2+
|0|0|A|3|
+6+0+-+-+
|2|3|5E4|
+-+-+-+-+

+-+-+-+-+
|   |   |
+-+-+-+-+
| |   | |
+ +-+-+ +
| |   | |
+-+-+-+-+
```

Sample Output

```
3
9A 123
123 5E4
3 5E4
```