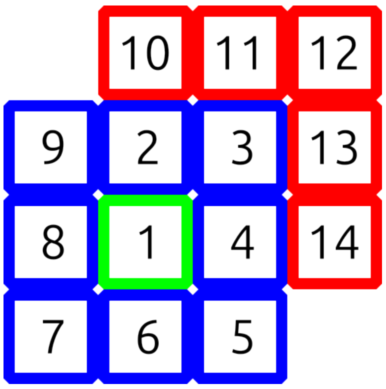
A. Alyona and a Square Jigsaw Puzzle

```
Input file:      standard input
Output file:     standard output
Time limit:      1 second
Memory limit:    256 megabytes
```

Alyona assembles an unusual square Jigsaw Puzzle. She does so in n days in the following manner:

- On the first day, she starts by placing the central piece in the center of the table.
- On each day after the first one, she places a certain number of pieces around the central piece in clockwise order, always finishing each square layer completely before starting a new one.

For example, she places the first 14 pieces in the following order:



The colors denote the layers. The third layer is still unfinished.

Alyona is happy if at the end of the day the assembled part of the puzzle does not have any started but unfinished layers. Given the number of pieces she assembles on each day, find the number of days Alyona is happy on.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line contains a single integer n ($1 \le n \le 100$), the number of days.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$, $a_1 = 1$), where a_i is the number of pieces Alyona assembles on the i-th day.

It is guaranteed in each test case that at the end of the n days, there are no unfinished layers.

Output

For each test case, print a single integer: the number of days when Alyona is happy.

Standard Input	Standard Output

5
1
1
2
1 8
5
1 3 2 1 2
7
	1
	2
	2
	2
	3

1 2 1 10 2 7 2
14
1 10 10 100 1 1 10 1 10 2 10 2 10 1	

Note

In the first test case, in the only day Alyona finishes the only layer.

In the second test case, on the first day, Alyona finishes the first layer, and on the second day, she finishes the second layer.

In the third test case, she finishes the second layer in a few days.

In the fourth test case, she finishes the second layer and immediately starts the next one on the same day, therefore, she is not happy on that day. She is only happy on the first and last days.

In the fifth test case, Alyona is happy on the first, fourth, and last days.

,

B. Replace Character

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

You're given a string s of length n, consisting of only lowercase English letters.

You must do the following operation exactly once:

- Choose any two indices i and j ($1 \le i, j \le n$). You can choose $i = j$.
- Set $s_i := s_j$.

You need to minimize the number of distinct permutations[†] of s. Output any string with the smallest number of distinct permutations after performing **exactly one** operation.

[†] A permutation of the string is an arrangement of its characters into any order. For example, "bac" is a permutation of "abc" but "bcc" is not.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first line of each test case contains n ($1 \le n \le 10$) — the length of string s.

The second line of each test case contains s of length n. The string contains only lowercase English letters.

Output

For each test case, output the required s after applying exactly one operation. If there are multiple solutions, print any of them.

Standard Input	Standard Output
6 3 abc 4 xyyx 8 alphabet 1 k 10 aabbccddee 6 ttbddq	cbc yyyx alphaaet k eabbccddee tttddq

Note

In the first test case, we can obtain the following strings in one operation: "abc", "bbc", "cbc", "aac", "acc", "aba", and "abb".

The string "`abc`" has 6 distinct permutations: "`abc`", "`acb`", "`bac`", "`bca`", "`cab`", and "`cba`".

The string "`cbc`" has 3 distinct permutations: "`bcc`", "`cbc`", and "`ccb`", which is the lowest of all the obtainable strings. In fact, all obtainable strings except "`abc`" have 3 permutations, so any of them would be accepted.

C. Swap Columns and Find a Path

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

There is a matrix consisting of 2 rows and n columns. The rows are numbered from 1 to 2 from top to bottom; the columns are numbered from 1 to n from left to right. Let's denote the cell on the intersection of the i-th row and the j-th column as (i, j). Each cell contains an integer; initially, the integer in the cell (i, j) is $a_{i,j}$.

You can perform the following operation any number of times (possibly zero):

- choose two columns and swap them (i. e. choose two integers x and y such that $1 \le x < y \le n$, then swap $a_{1,x}$ with $a_{1,y}$, and then swap $a_{2,x}$ with $a_{2,y}$).

After performing the operations, you have to choose a path from the cell $(1, 1)$ to the cell $(2, n)$. For every cell (i, j) in the path except for the last, the next cell should be either $(i + 1, j)$ or $(i, j + 1)$. Obviously, the path cannot go outside the matrix.

The *cost* of the path is the sum of all integers in all $(n + 1)$ cells belonging to the path. You have to perform the operations and choose a path so that its cost is **maximum** possible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 5000$). The description of the test cases follows.

Each test case consists of three lines:

- the first line contains one integer n ($1 \le n \le 5000$) — the number of columns in the matrix;
- the second line contains n integers $a_{1,1}, a_{1,2}, \ldots, a_{1,n}$ ($-10^5 \le a_{i,j} \le 10^5$) — the first row of the matrix;
- the third line contains n integers $a_{2,1}, a_{2,2}, \ldots, a_{2,n}$ ($-10^5 \le a_{i,j} \le 10^5$) — the second row of the matrix.

It is guaranteed that the sum of n over all test cases does not exceed 5000.

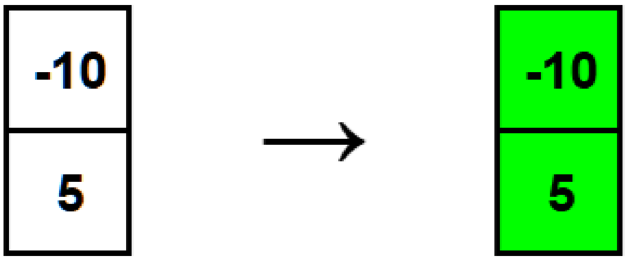
Output

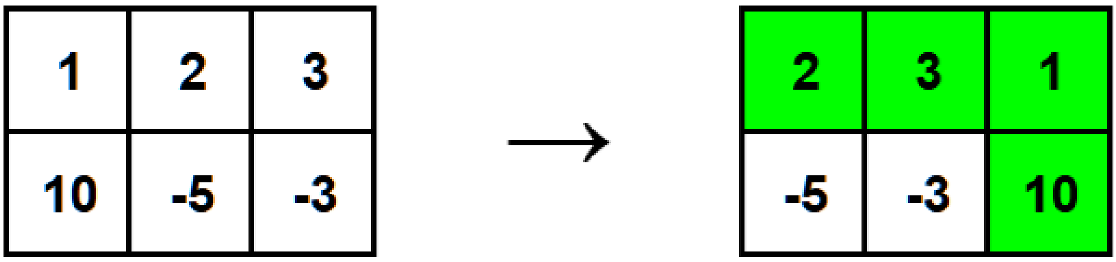
For each test case, print one integer — the maximum cost of a path you can obtain.

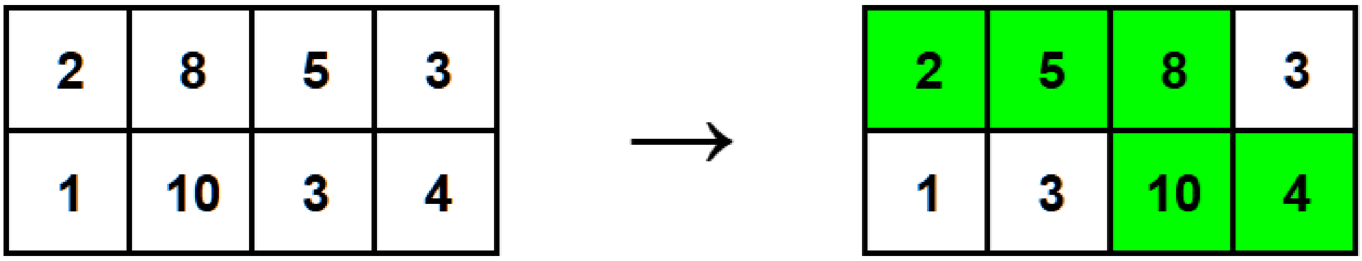
Standard Input	Standard Output
3	-5
1	16
-10	29
5	
3	
1 2 3	
10 -5 -3	
4	
2 8 5 3	
1 10 3 4	

Note

Here are the explanations of the first three test cases of the example. The left matrix is the matrix given in the input, the right one is the state of the matrix after several column swaps (possibly zero). The optimal path is highlighted in green.







,

D. Move Back at a Cost

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given an array of integers a of length n. You can perform the following operation zero or more times:

- In one operation choose an index i ($1 \le i \le n$), assign $a_i := a_i + 1$, and then move a_i to the back of the array (to the rightmost position). For example, if $a = [3, 5, 1, 9]$, and you choose $i = 2$, the array becomes $[3, 1, 9, 6]$.

Find the lexicographically smallest* array you can get by performing these operations.

*An array c is lexicographically smaller than an array d if and only if one of the following holds:

- c is a prefix of d, but $c \ne d$; or
- in the first position where c and d differ, the array c has a smaller element than the corresponding element in d.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line contains a single integer n ($1 \le n \le 10^5$), the length of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), the elements of the array.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, print the lexicographically smallest array you can get.

Standard Input	Standard Output

3
3
2 1 3
5
1 2 2 1 4
6
1 2 3 6 5 4
	1 3 3
	1 1 3 3 5
	1 2 3 4 6 7

,

E. Adventurers

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   256 megabytes
```

Once, four Roman merchants met in a Roman mansion to discuss their trading plans. They faced the following problem: they traded the same type of goods, and if they traded in the same city, they would inevitably incur losses. They decided to divide up the cities between them where they would trade.

The map of Rome can be represented in this problem as a plane with certain points marked — the cities of the Roman Empire.

The merchants decided to choose a certain *dividing point* (x_0, y_0). Then, in a city with coordinates (x_i, y_i),

- the first merchant sells goods if $x_0 \le x_i$ and $y_0 \le y_i$;
- the second merchant sells goods if $x_0 > x_i$ and $y_0 \le y_i$;
- the third merchant sells goods if $x_0 \le x_i$ and $y_0 > y_i$;
- the fourth merchant sells goods if $x_0 > x_i$ and $y_0 > y_i$.

The merchants want to choose (x_0, y_0) in such a way as to **maximize the smallest number of cities** that any of them gets (i. e., as fair as possible). Please find such a point for them.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($4 \le n \le 10^5$) — the number of cities on the map.

Each of the next n lines contains two integers x_i, y_i ($-10^9 \le x_i, y_i \le 10^9$) — the coordinates of the cities.

Note that some points may coincide. This is because some cities may be so close that they cannot be distinguished on the map at the given scale.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

Output

For each test case, in the first line, print a single integer k ($0 \le k \le \frac{n}{4}$) — the maximum possible number of cities that each merchant can get at a minimum.

In the second line, print two integers x_0 and y_0 ($|x_0|, |y_0| \le 10^9$) — the coordinates of the dividing point. If there are multiple suitable points, print any of them.

Standard Input	Standard Output

4
4
1 1
1 2
2 1
2 2
4
	1
	2 2
	0
	0 0
	2
	1 0

```
0 0
0 0
0 0
0 0
8
1 2
2 1
2 -1
1 -2
-1 -2
-2 -1
-2 1
-1 2
7
1 1
1 2
1 3
1 4
2 1
3 1
4 1
```

```
0
0 0
```

F. For the Emperor!

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

In Ancient Rome, a plan to defeat the barbarians was developed, but for its implementation, each city must be informed about it.

The northern part of the Roman Empire consists of n cities connected by m one-way roads. Initially, the i-th city has a_i messengers, and each messenger can freely move between cities following the existing roads. A messenger can carry a copy of the plan with him and inform the cities he visits, and can make unlimited copies for other messengers in the city he is currently in.

At the start, you will produce some number of plans and deliver them to messengers of your choice. Your goal is to make sure that every city is visited by a messenger with a plan. Find the smallest number of the plans you need to produce originally, so that the messengers will deliver them to every city, or determine that it is impossible to do so at all.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 100$). The description of the test cases follows.

The first line contains two integers n and m ($2 \le n \le 200$, $1 \le m \le 800$) — the number of cities and roads.

The second line contains n non-negative integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n$) — the initial number of messengers in each city.

Each of the following m lines contains two integers u and v ($1 \le u, v \le n, u \ne v$), indicating that there is a one-way road from city u to city v. The roads may repeat.

It is guaranteed that the sum of n over all test cases does not exceed 200. It is guaranteed that the sum of m over all test cases does not exceed 800.

Output

Output a single line containing a single integer — the smallest number of messengers you need to give a copy of the plan in the beginning, or -1 if it is not possible to inform all cities.

Standard Input	Standard Output
2	2
7 6	2
2 1 0 1 2 3 4	
1 2	
1 3	
2 4	
2 5	
3 6	
3 7	
4 4	
1 1 1 1	

1 2
1 3
2 4
3 4	