Problem A. Little Girl and Maximum Sum

Time Limit	1000 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

Topicwise Problem Link

The little girl loves the problems on array queries very much.

One day she came across a rather well-known problem: you've got an array of n elements (the elements of the array are indexed starting from 1); also, there are q queries, each one is defined by a pair of integers l_i, r_i $(1 \leq l_i \leq r_i \leq n)$. You need to find for each query the sum of elements of the array with indexes from l_i to r_i, inclusive.

The little girl found the problem rather boring. She decided to reorder the array elements before replying to the queries in a way that makes the sum of query replies maximum possible. Your task is to find the value of this maximum sum.

Input

The first line contains two space-separated integers n $(1 \leq n \leq 2 \cdot 10^5)$ and q $(1 \leq q \leq 2 \cdot 10^5)$ — the number of elements in the array and the number of queries, correspondingly.

The next line contains n space-separated integers a_i $(1 \leq a_i \leq 2 \cdot 10^5)$ — the array elements.

Each of the following q lines contains two space-separated integers l_i and r_i $(1 \leq l_i \leq r_i \leq n)$ — the i-th query.

Output

In a single line print, a single integer — the maximum sum of query replies after the array elements are reordered.

Please, do not use the `%lld` specifier to read or write 64-bit integers in C++. It is preferred to use the `cin`, `cout` streams or the `%I64d` specifier.

Examples

Input	Output
3 3 5 3 2 1 2 2 3 1 3	25

Input	Output
5 3 5 2 4 1 3 1 5 2 3 2 3	33

Problem B. Two TVs

Time Limit 2000 ms
Mem Limit 262144 kB

Topicwise Problem Link

Polycarp is a great fan of television.

He wrote down all the TV programs he is interested in for today. His list contains n shows, i-th of them starts at moment l_i and ends at moment r_i.

Polycarp owns two TVs. He can watch two different shows simultaneously with two TVs but he can only watch one show at any given moment on a single TV. If one show ends at the same moment some other show starts then you can't watch them on a single TV.

Polycarp wants to check out all n shows. Are two TVs enough to do so?

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of shows.

Each of the next n lines contains two integers l_i and r_i ($0 \le l_i < r_i \le 10^9$) — starting and ending time of i-th show.

Output

If Polycarp is able to check out all the shows using only two TVs then print "YES" (without quotes). Otherwise, print "NO" (without quotes).

Examples

Input	Output
3 1 2 2 3 4 5	YES

Input	Output
4 1 2 2 3 2 3 1 2	NO

Problem C. Restaurant Customers

Time Limit 1000 ms
Mem Limit 524288 kB

Topicwise Problem Link

You are given the arrival and leaving times of n customers in a restaurant.

What was the maximum number of customers in the restaurant at any time?

Input

The first input line has an integer n: the number of customers.

After this, there are n lines that describe the customers. Each line has two integers a and b : the arrival and leaving times of a customer.

You may assume that all arrival and leaving times are distinct.

Output

Print one integer: the maximum number of customers.

Constraints

- $1 \leq n \leq 2 \cdot 10^5$
- $1 \leq a < b \leq 10^9$

Example

Input	Output
3 5 8 2 4 3 9	2

Problem D. Karen and Coffee

Time Limit 2500 ms
Mem Limit 524288 kB

Topicwise Problem Link

To stay woke and attentive during classes, Karen needs some coffee!

Karen, a coffee aficionado, wants to know the optimal temperature for brewing the perfect cup of coffee. Indeed, she has spent some time reading several recipe books, including the universally acclaimed "The Art of the Covfefe".

She knows n coffee recipes. The i-th recipe suggests that coffee should be brewed between l_i and r_i degrees, inclusive, to achieve the optimal taste.

Karen thinks that a temperature is *admissible* if at least k recipes recommend it.

Karen has a rather fickle mind, and so she asks q questions. In each question, given that she only wants to prepare coffee with a temperature between a and b, inclusive, can you tell her how many admissible integer temperatures fall within the range?

Input

The first line of input contains three integers, n, k ($1 \le k \le n \le 200000$), and q ($1 \le q \le 200000$), the number of recipes, the minimum number of recipes a certain temperature must be recommended by to be admissible, and the number of questions Karen has, respectively.

The next n lines describe the recipes. Specifically, the i-th line among these contains two integers l_i and r_i ($1 \le l_i \le r_i \le 200000$), describing that the i-th recipe suggests that the coffee be brewed between l_i and r_i degrees, inclusive.

The next q lines describe the questions. Each of these lines contains a and b, ($1 \le a \le b \le 200000$), describing that she wants to know the number of admissible integer temperatures between a and b degrees, inclusive.

Output

For each question, output a single integer on a line by itself, the number of admissible integer temperatures between a and b degrees, inclusive.

Examples

Input	Output
3 2 4 91 94 92 97 97 99 92 94 93 97 95 96 90 100	3 3 0 4

Input	Output
2 1 1 1 1 200000 200000 90 100	0

Note

In the first test case, Karen knows 3 recipes.

1. The first one recommends brewing the coffee between 91 and 94 degrees, inclusive.

2. The second one recommends brewing the coffee between 92 and 97 degrees, inclusive.
3. The third one recommends brewing the coffee between 97 and 99 degrees, inclusive.

A temperature is *admissible* if at least 2 recipes recommend it.

She asks 4 questions.

In her first question, she wants to know the number of admissible integer temperatures between 92 and 94 degrees, inclusive. There are 3: 92, 93 and 94 degrees are all admissible.

In her second question, she wants to know the number of admissible integer temperatures between 93 and 97 degrees, inclusive. There are 3: 93, 94 and 97 degrees are all admissible.

In her third question, she wants to know the number of admissible integer temperatures between 95 and 96 degrees, inclusive. There are none.

In her final question, she wants to know the number of admissible integer temperatures between 90 and 100 degrees, inclusive. There are 4: 92, 93, 94 and 97 degrees are all admissible.

In the second test case, Karen knows 2 recipes.

1. The first one, "wikiHow to make Cold Brew Coffee", recommends brewing the coffee at exactly 1 degree.
2. The second one, "What good is coffee that isn't brewed at at least 36.3306 times the temperature of the surface of the sun?", recommends brewing the coffee at exactly 200000 degrees.

A temperature is *admissible* if at least 1 recipe recommends it.

In her first and only question, she wants to know the number of admissible integer temperatures that are actually reasonable. There are none.

Problem E. Greg and Array

Time Limit	1500 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

Topicwise Problem Link

Greg has an array $a = a_1, a_2, ..., a_n$ and m operations. Each operation looks as: l_i, r_i, d_i, $(1 \le l_i \le r_i \le n)$. To apply operation i to the array means to increase all array elements with numbers $l_i, l_i + 1, ..., r_i$ by value d_i.

Greg wrote down k queries on a piece of paper. Each query has the following form: x_i, y_i, $(1 \le x_i \le y_i \le m)$. That means that one should apply operations with numbers $x_i, x_i + 1, ..., y_i$ to the array.

Now Greg is wondering, what the array a will be after all the queries are executed. Help Greg.

Input

The first line contains integers n, m, k $(1 \le n, m, k \le 10^5)$. The second line contains n integers: $a_1, a_2, ..., a_n$ $(0 \le a_i \le 10^5)$ — the initial array.

Next m lines contain operations, the operation number i is written as three integers: l_i, r_i, d_i, $(1 \le l_i \le r_i \le n)$, $(0 \le d_i \le 10^5)$.

Next k lines contain the queries, the query number i is written as two integers: x_i, y_i, $(1 \le x_i \le y_i \le m)$.

The numbers in the lines are separated by single spaces.

Output

On a single line print n integers $a_1, a_2, \ldots, a_n$ — the array after executing all the queries. Separate the printed numbers by spaces.

Please, do not use the `%lld` specifier to read or write 64-bit integers in *C++*. It is preferred to use the `cin`, `cout` streams of the `%I64d` specifier.

Examples

Input	Output
3 3 3 1 2 3 1 2 1 1 3 2 2 3 4 1 2 1 3 2 3	9 18 17

Input	Output
1 1 1 1 1 1 1 1 1	2

Input	Output
4 3 6 1 2 3 4 1 2 1 2 3 2 3 4 4 1 2 1 3 2 3 1 2 1 3 2 3	5 18 31 20

Problem F. Queries about less or equal elements

Time Limit 2000 ms
Mem Limit 262144 kB

Topicwise Problem Link

You are given two arrays of integers a and b. For each element of the second array b_j you should find the number of elements in array a that are less than or equal to the value b_j.

Input

The first line contains two integers n, m ($1 \le n, m \le 2 \cdot 10^5$) — the sizes of arrays a and b.

The second line contains n integers — the elements of array a ($-10^9 \le a_i \le 10^9$).

The third line contains m integers — the elements of array b ($-10^9 \le b_j \le 10^9$).

Output

Print m integers, separated by spaces: the j-th of which is equal to the number of such elements in array a that are less than or equal to the value b_j.

Examples

Input	Output
5 4 1 3 5 7 9 6 4 2 8	3 2 1 4

Input	Output
5 5 1 2 1 2 5 3 1 4 1 5	4 2 4 2 5

Problem G. Inversion Count

Time Limit	3599 ms
Mem Limit	1572864 kB
Code Length Limit	50000 B
OS	Linux

Topicwise Problem Link

Let A[0 ... n - 1] be an array of n distinct positive integers. If i < j and A[i] > A[j] then the pair (i, j) is called an inversion of A. Given n and an array A your task is to find the number of inversions of A.

Input

The first line contains t, the number of testcases followed by a blank space. Each of the t tests start with a number n (n ≤ 200000). Then n + 1 lines follow. In the ith line a number A[i - 1] is given (A[i - 1] ≤ 10^7). The (n + 1)th line is a blank space.

Output

For every test output one line giving the number of inversions of A.

Example

Input	Output
2 3 3 1 2 5 2 3 8 6 1	2 5

Problem H. Sliding Window Median

Time Limit 1000 ms
Mem Limit 524288 kB

Topicwise Problem Link

You are given an array of n integers. Your task is to calculate the median of each window of k elements, from left to right.

The median is the middle element when the elements are sorted. If the number of elements is even, there are two possible medians and we assume that the median is the smaller of them.

Input

The first line contains two integers n and k: the number of elements and the size of the window.

Then there are n integers $x_1, x_2, \ldots, x_n$: the contents of the array.

Output

Print $n - k + 1$ values: the medians.

Constraints

- $1 \leq k \leq n \leq 2 \cdot 10^5$
- $1 \leq x_i \leq 10^9$

Example

Input	Output
8 3 2 4 3 5 8 1 2 1	3 4 5 5 2 1

Problem I. String Reversal

Time Limit 2000 ms
Mem Limit 262144 kB

Topicwise Problem Link

You are given a string s. You have to reverse it — that is, the first letter should become equal to the last letter before the reversal, the second letter should become equal to the second-to-last letter before the reversal — and so on. For example, if your goal is to reverse the string "abddea", you should get the string "aeddba". To accomplish your goal, you can swap the **neighboring elements of the string**.

Your task is to calculate the minimum number of swaps you have to perform to reverse the given string.

Input

The first line contains one integer n ($2 \le n \le 200\,000$) — the length of s.

The second line contains s — a string consisting of n lowercase Latin letters.

Output

Print one integer — the minimum number of swaps of neighboring elements you have to perform to reverse the string.

Examples

Input	Output
5 aaaza	2

Input	Output
6 cbaabc	0

Input	Output
9 icpcsguru	30

Note

In the first example, you have to swap the third and the fourth elements, so the string becomes "aazaa". Then you have to swap the second and the third elements, so the string becomes "azaaa". So, it is possible to reverse the string in two swaps.

Since the string in the second example is a palindrome, you don't have to do anything to reverse it.

Problem J. Maximum Crossings (Easy Version)

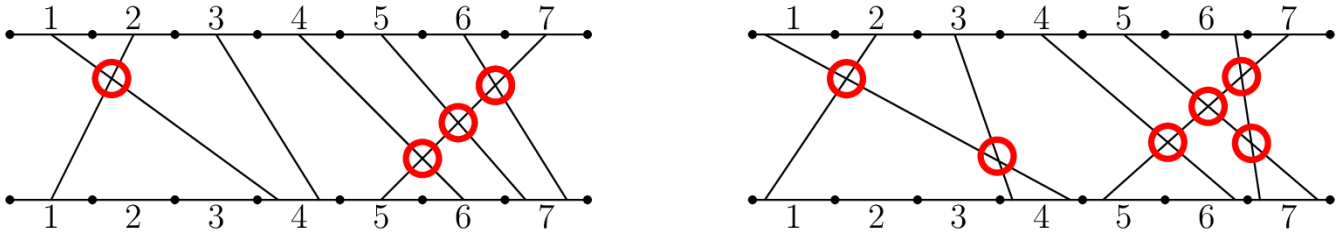
Time Limit 1000 ms
Mem Limit 262144 kB

Topicwise Problem Link

The only difference between the two versions is that in this version $n \le 1000$ and the sum of n over all test cases does not exceed 1000.

A *terminal* is a row of n equal segments numbered 1 to n in order. There are two terminals, one above the other.

You are given an array a of length n. For all $i = 1, 2, \ldots, n$, there should be a straight wire from some point on segment i of the top terminal to some point on segment a_i of the bottom terminal. You can't select the endpoints of a segment. For example, the following pictures show two possible wirings if $n = 7$ and $a = [4, 1, 4, 6, 7, 7, 5]$.



A *crossing* occurs when two wires share a point in common. In the picture above, crossings are circled in red.

What is the **maximum** number of crossings there can be if you place the wires optimally?

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains an integer n ($1 \le n \le 1000$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

The sum of n across all test cases does not exceed 1000.

Output

For each test case, output a single integer — the **maximum** number of crossings there can be if you place the wires optimally.

Examples

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

Note

The first test case is shown in the second picture in the statement.

In the second test case, the only wiring possible has the two wires cross, so the answer is 1.

In the third test case, the only wiring possible has one wire, so the answer is 0.

Problem K. Maximum Crossings (Hard Version)

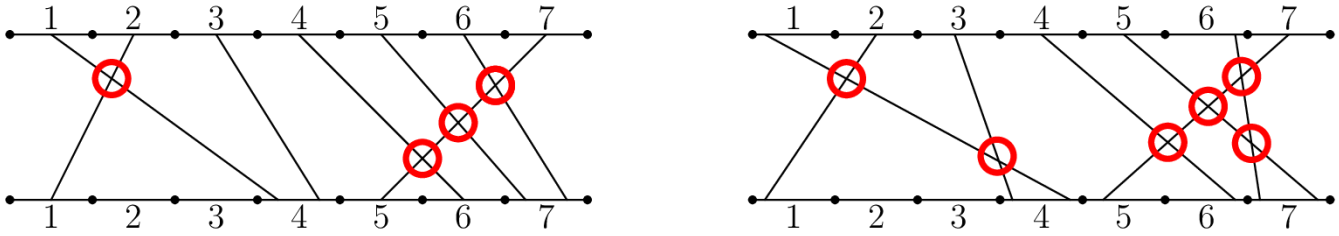
Time Limit 1000 ms
Mem Limit 262144 kB

Topicwise Problem Link

The only difference between the two versions is that in this version $n \le 2 \cdot 10^5$ and the sum of n over all test cases does not exceed $2 \cdot 10^5$.

A *terminal* is a row of n equal segments numbered 1 to n in order. There are two terminals, one above the other.

You are given an array a of length n. For all $i = 1, 2, \ldots, n$, there should be a straight wire from some point on segment i of the top terminal to some point on segment a_i of the bottom terminal. You can't select the endpoints of a segment. For example, the following pictures show two possible wirings if $n = 7$ and $a = [4, 1, 4, 6, 7, 7, 5]$.



A *crossing* occurs when two wires share a point in common. In the picture above, crossings are circled in red.

What is the **maximum** number of crossings there can be if you place the wires optimally?

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The first line of each test case contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

The sum of n across all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer — the **maximum** number of crossings there can be if you place the wires optimally.

Examples

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

Note

The first test case is shown in the second picture in the statement.

In the second test case, the only wiring possible has the two wires cross, so the answer is 1.

In the third test case, the only wiring possible has one wire, so the answer is 0.

Problem L. Enemy is weak

Time Limit	5000 ms
Mem Limit	262144 kB
Input File	stdin
Output File	stdout

Topicwise Problem Link

The Romans have attacked again. This time they are much more than the Persians but Shapur is ready to defeat them. He says: "A lion is never afraid of a hundred sheep".

Nevertheless Shapur has to find weaknesses in the Roman army to defeat them. So he gives the army a weakness number.

In Shapur's opinion the weakness of an army is equal to the number of triplets i, j, k such that $i < j < k$ and $a_i > a_j > a_k$ where a_x is the power of man standing at position x. The Roman army has one special trait — powers of all the people in it are distinct.

Help Shapur find out how weak the Romans are.

Input

The first line of input contains a single number n ($3 \le n \le 10^6$) — the number of men in Roman army. Next line contains n different positive integers a_i ($1 \le i \le n$, $1 \le a_i \le 10^9$) — powers of men in the Roman army.

Output

A single integer number, the weakness of the Roman army.

Please, do not use `%lld` specificator to read or write 64-bit integers in C++. It is preffered to use `cout` (also you may use `%I64d`).

Examples

Input	Output
3 3 2 1	1

Input	Output
3 2 3 1	0

Input	Output
4 10 8 3 1	4

Input	Output
4 1 5 4 3	1

Problem M. Pair of Topics

Time Limit 2000 ms
Mem Limit 262144 kB

Topicwise Problem Link

The next lecture in a high school requires two topics to be discussed. The i-th topic is interesting by a_i units for the teacher and by b_i units for the students.

The pair of topics i and j ($i < j$) is called **good** if $a_i + a_j > b_i + b_j$ (i.e. it is more interesting for the teacher).

Your task is to find the number of **good** pairs of topics.

Input

The first line of the input contains one integer n ($2 \le n \le 2 \cdot 10^5$) — the number of topics.

The second line of the input contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the interestingness of the i-th topic for the teacher.

The third line of the input contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 10^9$), where b_i is the interestingness of the i-th topic for the students.

Output

Print one integer — the number of **good** pairs of topic.

Examples

Input	Output
5 4 8 2 6 2 4 5 4 1 3	7

Input	Output
4 1 3 2 4 1 3 2 4	0

Problem N. Greetings

Time Limit 5000 ms
Mem Limit 262144 kB

Topicwise Problem Link

There are n people on the number line; the i-th person is at point a_i and wants to go to point b_i. For each person, $a_i < b_i$, and the starting and ending points of all people are distinct. (That is, all of the $2n$ numbers $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ are distinct.)

All the people will start moving simultaneously at a speed of 1 unit per second until they reach their final point b_i. When two people meet at the same point, they will greet each other once. How many greetings will there be?

Note that a person can still greet other people even if they have reached their final point.

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. The description of test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of people.

Then n lines follow, the i-th of which contains two integers a_i and b_i ($-10^9 \le a_i < b_i \le 10^9$) — the starting and ending positions of each person.

For each test case, all of the $2n$ numbers $a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_n$ are distinct.

The sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single integer denoting the number of greetings that will happen.

Examples

Input	Output
5 2 2 3 1 4 6 2 6 3 9 4 5 1 8 7 10 -2 100 4 -10 10 -5 5 -12 12 -13 13 5 -4 9 -2 5 3 4 6 7 8 10 4 1 2 3 4 5 6 7 8	1 9 6 4 0

Note

In the first test case, the two people will meet at point 3 and greet each other.

Problem O. Luntik and Subsequences

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Luntik came out for a morning stroll and found an array a of length n. He calculated the sum s of the elements of the array ($s = \sum_{i=1}^{n} a_i$). Luntik calls a subsequence of the array a *nearly full* if the sum of the numbers in that subsequence is equal to $s - 1$.

Luntik really wants to know the number of *nearly full* subsequences of the array a. But he needs to come home so he asks you to solve that problem!

A sequence x is a subsequence of a sequence y if x can be obtained from y by deletion of several (possibly, zero or all) elements.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. The next $2 \cdot t$ lines contain descriptions of test cases. The description of each test case consists of two lines.

The first line of each test case contains a single integer n ($1 \le n \le 60$) — the length of the array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the elements of the array a.

Output

For each test case print the number of *nearly full* subsequences of the array.

Examples

Input	Output
5 5 1 2 3 4 5 2 1000 1000 2 1 0 5 3 0 2 1 1 5 2 1 0 3 0	1 0 2 4 4

Note

In the first test case, $s = 1 + 2 + 3 + 4 + 5 = 15$, only $(2, 3, 4, 5)$ is a nearly full subsequence among all subsequences, the sum in it is equal to $2 + 3 + 4 + 5 = 14 = 15 - 1$.

In the second test case, there are no nearly full subsequences.

In the third test case, $s = 1 + 0 = 1$, the nearly full subsequences are (0) and $()$ (the sum of an empty subsequence is 0).

Problem P. Social Distance

Time Limit 1500 ms
Mem Limit 262144 kB

Problem Link

m chairs are arranged in a circle sequentially. The chairs are numbered from 0 to $m - 1$. n people want to sit in these chairs. The i-th of them wants at least $a[i]$ empty chairs both on his right and left side.

More formally, if the i-th person sits in the j-th chair, then no one else should sit in the following chairs: $(j - a[i]) \bmod m$, $(j - a[i] + 1) \bmod m$, ... $(j + a[i] - 1) \bmod m$, $(j + a[i]) \bmod m$.

Decide if it is possible to sit down for all of them, under the given limitations.

Input

The input consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 5 \cdot 10^4$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n and m ($2 \le n \le 10^5$, $1 \le m \le 10^9$) — the number of people and the number of chairs.

The next line contains n integers, a_1, a_2, ... a_n ($1 \le a_i \le 10^9$) — the minimum number of empty chairs, on both sides of the i-th person.

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

Output

For each test case print "YES" (without quotes) if it is possible for everyone to sit down and fulfil the restrictions, and "NO" (without quotes) otherwise.

You may print every letter in any case you want (so, for example, the strings "yEs", "yes", "Yes" and "YES" will all be recognized as positive answers).

Examples

Input	Output
6 3 2 1 1 1 2 4 1 1 2 5 2 1 3 8 1 2 1 4 12 1 2 1 3 4 19 1 2 1 3	NO YES NO YES NO YES

Note

Test case 1: $n > m$, so they can not sit down.

Test case 2: the first person can sit 2-nd and the second person can sit in the 0-th chair. Both of them want at least 1 empty chair on both sides, chairs 1 and 3 are free, so this is a good solution.

Test case 3: if the second person sits down somewhere, he needs 2 empty chairs, both on his right and on his left side, so it is impossible to find a place for the first person, because there are only 5 chairs.

Test case 4: they can sit in the 1-st, 4-th, 7-th chairs respectively.

Problem Q. Mystic Permutation

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

Monocarp is a little boy who lives in Byteland and he loves programming.

Recently, he found a permutation of length n. He has to come up with a *mystic* permutation. It has to be a new permutation such that it differs from the old one in each position.

More formally, if the old permutation is $p_1, p_2, \ldots, p_n$ and the new one is $q_1, q_2, \ldots, q_n$ it must hold that

$$p_1 \neq q_1, p_2 \neq q_2, \ldots, p_n \neq q_n.$$

Monocarp is afraid of lexicographically large permutations. Can you please help him to find the lexicographically minimal *mystic* permutation?

Input

There are several test cases in the input data. The first line contains a single integer t ($1 \leq t \leq 200$) — the number of test cases. This is followed by the test cases description.

The first line of each test case contains a positive integer n ($1 \leq n \leq 1000$) — the length of the permutation.

The second line of each test case contains n distinct positive integers $p_1, p_2, \ldots, p_n$ ($1 \leq p_i \leq n$). It's guaranteed that p is a permutation, i. e. $p_i \neq p_j$ for all $i \neq j$.

It is guaranteed that the sum of n does not exceed 1000 over all test cases.

Output

For each test case, output n positive integers — the lexicographically minimal mystic permutations. If such a permutation does not exist, output -1 instead.

Examples

Input	Output
4 3 1 2 3 5 2 3 4 5 1 4 2 3 1 4 1 1	2 3 1 1 2 3 4 5 1 2 4 3 -1

Note

In the first test case possible permutations that are mystic are $[2, 3, 1]$ and $[3, 1, 2]$. Lexicographically smaller of the two is $[2, 3, 1]$.

In the second test case, $[1, 2, 3, 4, 5]$ is the lexicographically minimal permutation and it is also mystic.

In third test case possible mystic permutations are $[1, 2, 4, 3]$, $[1, 4, 2, 3]$, $[1, 4, 3, 2]$, $[3, 1, 4, 2]$, $[3, 2, 4, 1]$, $[3, 4, 2, 1]$, $[4, 1, 2, 3]$, $[4, 1, 3, 2]$ and $[4, 3, 2, 1]$. The smallest one is $[1, 2, 4, 3]$.

Problem R. Shuffle Hashing

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

Polycarp has built his own web service. Being a modern web service it includes login feature. And that always implies password security problems.

Polycarp decided to store the hash of the password, generated by the following algorithm:

1. take the password p, consisting of lowercase Latin letters, and shuffle the letters randomly in it to obtain p' (p' can still be equal to p);
2. generate two random strings, consisting of lowercase Latin letters, s_1 and s_2 (any of these strings can be empty);
3. the resulting hash $h = s_1 + p' + s_2$, where addition is string concatenation.

For example, let the password $p =$ "abacaba". Then p' can be equal to "aabcaab". Random strings $s1 =$ "zyx" and $s2 =$ "kjh". Then $h =$ "zyxaabcaabkjh".

Note that no letters could be deleted or added to p to obtain p', only the order could be changed.

Now Polycarp asks you to help him to implement the password check module. Given the password p and the hash h, check that h can be the hash for the password p.

Your program should answer t independent test cases.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

The first line of each test case contains a non-empty string p, consisting of lowercase Latin letters. The length of p does not exceed 100.

The second line of each test case contains a non-empty string h, consisting of lowercase Latin letters. The length of h does not exceed 100.

Output

For each test case print the answer to it — "YES" if the given hash h could be obtained from the given password p or "NO" otherwise.

Examples

Input	Output
5 abacaba zyxaabcaabkjh onetwothree threetwoone one zzonneyy one none twenty ten	YES YES NO YES NO

Note

The first test case is explained in the statement.

In the second test case both s_1 and s_2 are empty and $p' =$ "threetwoone" is p shuffled.

In the third test case the hash could not be obtained from the password.

In the fourth test case $s_1 =$ "n", s_2 is empty and $p' =$ "one" is p shuffled (even thought it stayed the same).

In the fifth test case the hash could not be obtained from the password.

Problem S. Longest Palindrome

Time Limit 1000 ms
Mem Limit 262144 kB

Problem Link

Returning back to problem solving, Gildong is now studying about palindromes. He learned that a *palindrome* is a string that is the same as its reverse. For example, strings "pop", "noon", "x", and "kkkkkk" are palindromes, while strings "moon", "tv", and "abab" are not. **An empty string is also a palindrome.**

Gildong loves this concept so much, so he wants to play with it. He has n *distinct* strings of equal length m. He wants to discard some of the strings (possibly none or all) and reorder the remaining strings so that the concatenation becomes a palindrome. He also wants the palindrome to be as long as possible. Please help him find one.

Input

The first line contains two integers n and m ($1 \leq n \leq 100, 1 \leq m \leq 50$) — the number of strings and the length of each string.

Next n lines contain a string of length m each, consisting of lowercase Latin letters only. All strings are *distinct*.

Output

In the first line, print the length of the longest palindrome string you made.

In the second line, print that palindrome. If there are multiple answers, print any one of them. If the palindrome is empty, print an empty line or don't print this line at all.

Examples

Input	Output
3 3 tab one bat	6 tabbat

Input	Output
4 2 oo ox xo xx	6 oxxxxo

Input	Output
3 5 hello codef orces	0

Input	Output
9 4 abab baba abcd bcde cdef defg wxyz zyxw ijji	20 ababwxyzijjizyxwbaba

Note

In the first example, "`battab`" is also a valid answer.

In the second example, there can be 4 different valid answers including the sample output. We are not going to provide any hints for what the others are.

In the third example, the empty string is the only valid palindrome string.

Problem T. Inaccurate Subsequence Search

Time Limit 2000 ms
Mem Limit 262144 kB

Problem Link

Maxim has an array a of n integers and an array b of m integers ($m \le n$).

Maxim considers an array c of length m to be good if the elements of array c can be rearranged in such a way that at least k of them match the elements of array b.

For example, if $b = [1, 2, 3, 4]$ and $k = 3$, then the arrays $[4, 1, 2, 3]$ and $[2, 3, 4, 5]$ are good (they can be reordered as follows: $[1, 2, 3, 4]$ and $[5, 2, 3, 4]$), while the arrays $[3, 4, 5, 6]$ and $[3, 4, 3, 4]$ are not good.

Maxim wants to choose every subsegment of array a of length m as the elements of array c. Help Maxim count how many selected arrays will be good.

In other words, find the number of positions $1 \le l \le n - m + 1$ such that the elements $a_l, a_{l+1}, \ldots, a_{l+m-1}$ form a good array.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains three integers n, m, and k ($1 \le k \le m \le n \le 2 \cdot 10^5$) — the number of elements in arrays a and b, the required number of matching elements.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the elements of array a. Elements of the array a are not necessarily unique.

The third line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^6$) — the elements of array b. Elements of the array b are not necessarily unique.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$. Similarly, it is guaranteed that the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the number of good subsegments of array a on a separate line.

Examples

Input	Output
5 7 4 2 4 1 2 3 4 5 6 1 2 3 4 7 4 3 4 1 2 3 4 5 6 1 2 3 4 7 4 4 4 1 2 3 4 5 6 1 2 3 4 11 5 3 9 9 2 2 10 9 7 6 3 6 3 6 9 7 8 10 4 1 1 4 1 5 6 6	4 3 2 4 1

Note

In the first example, all subsegments are good.

In the second example, good subsegments start at positions 1, 2, and 3.

In the third example, good subsegments start at positions 1 and 2.