Kotlin Heroes: Episode 1

A. Three Integers Again

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

We have three **positive** integers a, b and c. You don't know their values, but you know some information about them. Consider all three pairwise sums, i.e. the numbers $a + b$, $a + c$ and $b + c$. You know exactly two (any) of three pairwise sums.

Your task is to find such three **positive** integers a, b and c which match the given information. It means that if you consider $a + b$, $a + c$ and $b + c$, two out of all three of them are given in the input. Among all such triples, you must choose one with the **minimum possible sum** $a + b + c$, and among all triples with the minimum sum, you can print any.

You have to process q independent queries.

Input

The first line of the input contains one integer q ($1 \le q \le 1000$) — the number of queries.

The next q lines contain queries. Each query is given as two integers x and y ($2 \le x, y \le 2 \cdot 10^9$), where x and y are any two out of the three numbers $a + b$, $a + c$ and $b + c$.

Output

For each query print the answer to it: three **positive** integers a, b and c consistent with the given information. Among all such triples, you have to choose one with the **minimum possible sum** $a + b + c$. Among all triples with the minimum sum, you can print any.

Example

input
3 123 13 2 2 2000000000 2000000000
output
111 1 12 1 1 1 1999999999 1 1

B. Bad Days

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Every day Kotlin heroes analyze the statistics of their website. For n days, they wrote out n numbers $a_1, a_2, \ldots, a_n$, where a_i is the number of visits on the i-th day.

They believe that a day is bad if there are at least 2 days before it with a strictly greater number of visits. For example, if $n = 8$ and $a = [3, 1, 4, 1, 5, 9, 2, 6]$, then the day 4 is bad (because $a_4 = 1$, but there are $a_1 = 3$ and $a_3 = 4$). Also, the day with the number 7 is bad too.

Write a program that finds the number of bad days.

Input

The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$), where n is the number of days. The second line contains n positive integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), where a_i is the number of website visits on the i-th day.

Output

Print the number of bad days, i.e. such days that there are at least two days before it with a strictly greater number of visits.

Examples

input
8 3 1 4 1 5 9 2 6
output

2

input
5 1 1 1 1 1
output
0

input
13 2 7 1 8 2 8 1 8 2 8 4 5 9
output
6

C. Minus and Minus Give Plus

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Everyone knows that two consecutive (adjacent) "minus" signs can be replaced with a single "plus" sign.

You are given the string s, consisting of "plus" and "minus" signs only. Zero or more operations can be performed with it. In each operation you can choose any two adjacent "minus" signs, and replace them with a single "plus" sign. Thus, in one operation, the length of the string is reduced by exactly 1.

You are given two strings s and t. Determine if you can use 0 or more operations to get the string t from the string s.

Input

The first line of the input contains an integer k ($1 \le k \le 10^5$), denoting the number of test cases in the input. The following lines contain descriptions of the test sets, each set consists of two lines. First comes the line containing s (the length of the line s does not exceed $2 \cdot 10^5$), then comes the line containing t (the length of the line t does not exceed $2 \cdot 10^5$). The lines s and t are non-empty, and they contain only "plus" and "minus" signs.

The sum of the lengths of lines s over all test cases in the input does not exceed $2 \cdot 10^5$. Similarly, the sum of the lengths of lines t over all test cases in the input does not exceed $2 \cdot 10^5$.

Output

Print k lines: the i-th line must contain YES if the answer to the i-th test case is positive, otherwise NO. Print YES and NO using uppercase letters only.

Example

input
5 -+--+ -+++ -------- -+--+- - + -- --- +++ +++
output
YES YES NO NO YES

D. Decoding of Integer Sequences

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp is developing a method for transmitting n integer sequences over a network. This method should support the transmission of an arbitrary number of integer sequences; sequences can have different lengths. The sequences contain arbitrary **non-negative** integers.

Polycarp developed the following encoding procedure:

- We assume that the sequences are numbered from 1 to n.
- We add a special terminating marker to each sequence at the end, an element equal to -1.
- The encoding result is a new single sequence that contains all the elements of the specified n in a special order: first, add to the result of coding all the first elements of the sequences (in the order from the 1-st to the n-th), then all the second elements (in the order from the 1-st to the n-th) and so on, if the sequence does not have the corresponding element, then it is simply skipped. The process ends when all elements of all sequences are added.

For example, if you want to encode three sequences $[3, 1, 4]$, $[2, 7]$ and $[1, 2, 3, 4]$, then the sequence of actions will be as follows:

- we modify all three sequences by appending -1: $[3, 1, 4, -1]$, $[2, 7, -1]$ and $[1, 2, 3, 4, -1]$;
- we write out all the first elements, we get $[3, 2, 1]$;
- then write down all the second elements, we get $[3, 2, 1, 1, 7, 2]$;
- then write down all the third elements, we get $[3, 2, 1, 1, 7, 2, 4, -1, 3]$;
- then write down all fourth elements, get $[3, 2, 1, 1, 7, 2, 4, -1, 3, -1, 4]$ (note that the second sequence has already ended);
- then write down all the fifth elements, we get $[3, 2, 1, 1, 7, 2, 4, -1, 3, -1, 4, -1]$ (note that the first and second sequences have already ended);
- all the sequences are ended now, and the encoding process is finished;
- the encoding result is: $[3, 2, 1, 1, 7, 2, 4, -1, 3, -1, 4, -1]$.

Your task is to implement decoding by a given encoding result.

Input

The first line contains integer number m ($1 \le m \le 3 \cdot 10^5$), denoting the length of the encoding result. The second line contains the result of encoding as a sequence of integers $b_1, b_2, \ldots, b_m$ ($-1 \le b_i \le 100$).

It is guaranteed that in the initial sequences before encoding contains only non-negative integers from 0 to 100, that you are in fact given the result of correct encoding (in other words, it is guaranteed that the answer exists). It is possible that one or more initial sequences were empty before encoding.

Output

Print n, where n is the number of encoded sequences. Then print n lines in the format $k_i, a_{i1}, a_{i2}, \ldots, a_{ik_i}$, where k_i is the length of the i-th sequence, and $a_{i1}, a_{i2}, \ldots, a_{ik_i}$ are its elements. Separate the numbers in the lines with spaces. Please note that the encoding procedure is such that every possible encoding result can be decoded in only one way.

Examples

input
12 3 2 1 1 7 2 4 -1 3 -1 4 -1
output
3 3 3 1 4 2 2 7 4 1 2 3 4

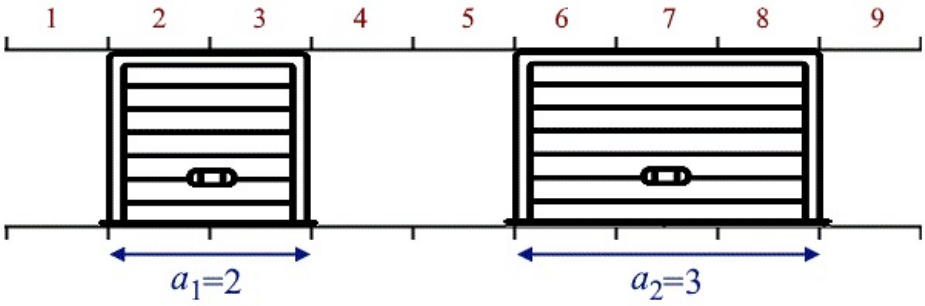
input
6 2 -1 2 -1 3 -1
output
3 1 2 0 2 2 3

E. Sliding Doors

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Imagine that you are the CEO of a big old-fashioned company. Unlike any modern and progressive company (such as JetBrains), your company has a dress code. That's why you have already allocated a spacious room for your employees where they can change their clothes. Moreover, you've already purchased an m-compartment wardrobe, so the i-th employee can keep his/her belongings in the i-th cell (of course, all compartments have equal widths).

The issue has occurred: the wardrobe has sliding doors! More specifically, the wardrobe has n doors (numbered from left to right) and the j-th door has width equal to a_j wardrobe's cells. The wardrobe has single rails so that no two doors can slide past each other.



Extremely schematic example of a wardrobe: $m = 9$, $n = 2$, $a_1 = 2$, $a_2 = 3$.

The problem is as follows: sometimes to open some cells you must close some other cells (since all doors are placed on the single track). For example, if you have a 4-compartment wardrobe (i.e. $m = 4$) with $n = 2$ one-cell doors (i.e. $a_1 = a_2 = 1$) and you need to open the 1-st and the 3-rd cells, you have to close the 2-nd and the 4-th cells.

As CEO, you have a complete schedule for the next q days. Now you are wondering: is it possible that all employees who will come on the k-th day can access their cells simultaneously?

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 4 \cdot 10^5$) — the number of doors and compartments respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le m$, $\sum a_i \le m$) — the corresponding widths of the doors.

The third line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of days you have to process.

The next q lines describe schedule of each day. Each schedule is represented as an integer c_k followed by c_k integers $w_1, w_2, \ldots, w_{c_k}$ ($1 \le c_k \le 2 \cdot 10^5$, $1 \le w_1 < w_2 < \cdots < w_{c_k} \le m$) — the number of employees who will come on the k-th day, and their indices in ascending order.

It's guaranteed that $\sum c_k$ doesn't exceed $2 \cdot 10^5$.

Output

Print q answers. Each answer is "YES" or "NO" (case insensitive). Print "YES" if it is possible, that all employees on the corresponding day can access their compartments simultaneously.

Example

input
3 10 2 3 2 6 1 5 2 1 10 2 2 9 2 5 6 3 1 7 8 4 1 2 3 4
output
YES YES NO NO YES NO

F. Wheels

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has n wheels and a car with m slots for wheels. The initial pressure in the i-th wheel is a_i.

Polycarp's goal is to take **exactly m wheels** among the given n wheels and equalize the pressure in them (then he can put these wheels in a car and use it for driving). In one minute he can decrease or increase the pressure in any (single) wheel by 1. He can increase the pressure **no more than k times** in total because it is hard to pump up wheels.

Help Polycarp and say what is the minimum number of minutes he needs to spend to equalize the pressure of at least m wheels among the given n wheels.

Input

The first line of the input contains three integers n, m and k ($1 \le m \le n \le 2 \cdot 10^5, 0 \le k \le 10^9$) — the number of wheels, the number of slots for wheels in a car and the number of times Polycarp can increase by 1 the pressure in a wheel.

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 pressure in the i-th wheel.

Output
Print one integer — the minimum number of minutes Polycarp needs to spend to equalize the pressure in at least m wheels among the given n wheels.

Examples

input
6 6 7 6 15 16 20 1 5
output
39

input
6 3 1 4 8 15 16 23 42
output
8

input
5 4 0 5 5 5 4 5
output
0

G. Graph Decomposition

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph consisting of n vertices and m edges.

Recall that a cycle is a path that starts and ends in the same vertex. A cycle in a graph is called simple if it contains each vertex (except the starting and ending one) no more than once (the starting and the ending one is contained always **twice**). Note that loops are considered to be simple cycles.

In one move you can choose **any** simple cycle in this graph and erase the edges corresponding to this cycle (corresponding vertices remain in the graph). It is **allowed** to erase the loop or two copies of the same edge (take a look at examples).

Your problem is to apply some sequence of moves to obtain the graph without edges. **It is not necessary to minimize the number of cycles**. If it is impossible, print "NO".

Input
The first line of the input contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of vertices and the number of edges in the graph.

The next m lines contain edges of the graph. The i-th line contains the i-th edge x_i, y_i ($1 \le x_i, y_i \le n$), where x_i and y_i are vertices connected by the i-th edge. The graph **can contain loops or multiple edges**.

Output
If it is impossible to decompose the given graph into simple cycles, print "NO" in the first line.

Otherwise print "YES" in the first line. In the second line print k — the number of simple cycles in the graph decomposition.

In the next k lines print cycles themselves. The j-th line should contain the j-th cycle. First, print c_j — the number of vertices in the j-th cycle. Then print the cycle as a sequence of vertices. All neighbouring (adjacent) vertices in the printed path should be connected by an edge that isn't contained in other cycles.

Examples

input
6 9 1 2 2 3 1 3 2 4 2 5 4 5 3 5 3 6 5 6
output
YES 3

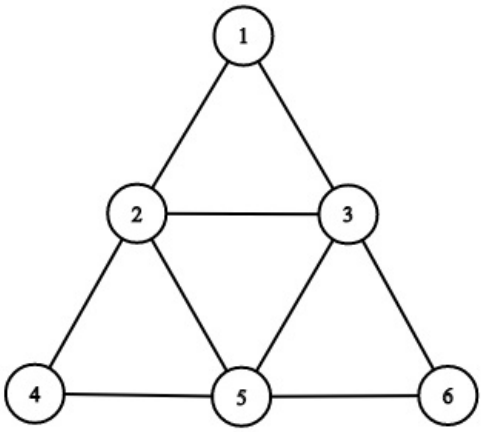
```
4 2 5 4 2
4 3 6 5 3
4 1 3 2 1
```

input
4 7 1 1 1 2 2 3 3 4 4 1 1 3 1 3
output
YES 3 2 1 1 5 1 4 3 2 1 3 1 3 1

input
4 8 1 1 1 2 2 3 3 4 4 1 2 4 1 3 1 3
output
NO

Note

The picture corresponding to the first example:



H. Longest Saw

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given the sequence $a_1, a_2, \ldots, a_n$. You can choose any subset of elements and then reorder them to create a "saw".

The sequence $b_1, b_2, \ldots, b_m$ is called a "saw" if the elements satisfy one of the following series of inequalities:
$b_1 > b_2 < b_3 > b_4 < \ldots$ or $b_1 < b_2 > b_3 < b_4 > \ldots$.

Find the longest saw which can be obtained from a given array.

Note that both the given sequence a and the required saw b can contain duplicated (non-unique) values.

Input

The first line contains an integer t ($1 \le t \le 10^5$) — the number of test cases in the input. Then the descriptions of the t test cases follow. Each test case begins with a line containing integer n ($1 \le n \le 2 \cdot 10^5$). Then a line containing n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) follows.

It's guaranteed that $\sum n$ doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print two lines: print the length of the longest saw in the first line, and the saw itself in the second line. If there are several solutions, print any of them.

Example

input
3 10 10 9 8 7 6 5 4 3 2 1 7 1 2 2 2 3 2 2 3 100 100 100
output
10 1 6 2 7 3 8 4 9 5 10 4 2 1 3 2 1 100

I. Good Subsets

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n segments on the Ox axis. The i-th segment is given as a pair l_i, r_i, where l_i is the position of the left end of the i-th segment and r_i is the position of the right end of the i-th segment. Segments may intersect, overlap, or even coincide. A segment is a set of numbers (including floating-point numbers) lying between the segment ends or coinciding with them. Formally, the segment $[l, r] = \{x \mid x \in \mathbb{R},\ l \le x \le r\}$.

Let the *union of segments* be the set of all axis points covered by the set of segments. Let's call a subset of the given segments **good** if its *union* equals the *union* of all n segments.

Your task is to calculate the number of **good** subsets of the given n segments. Since the answer may be very large, print it modulo 998244353.

Input

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

The next n lines contain segments. The i-th segment is given as a pair l_i, r_i ($1 \le l_i \le r_i \le 10^9$), where l_i is the left border of the segment and r_i is the right border of the segment. Segments may intersect, overlap, or even coincide.

Output

Print the number of **good** subsets of the given set of segments. Since the answer may be very large, print it modulo 998244353.

Examples

input
3 1 1 2 6 1 6
output
4

input
2 3 4 2 5
output
2

input
4 1 2 5 5 2 3 1 3
output
5

The only programming contests Web 2.0 platform