Educational Codeforces Round 79 (Rated for Div. 2)

A. New Year Garland

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

Polycarp is sad — New Year is coming in few days but there is still no snow in his city. To bring himself New Year mood, he decided to decorate his house with some garlands.

The local store introduced a new service this year, called "Build your own garland". So you can buy some red, green and blue lamps, provide them and the store workers will solder a single garland of them. The resulting garland will have all the lamps you provided put in a line. Moreover, no pair of lamps of the same color will be adjacent to each other in this garland!

For example, if you provide 3 red, 3 green and 3 blue lamps, the resulting garland can look like this: "RGBRBGBGR" ("RGB" being the red, green and blue color, respectively). Note that it's ok to have lamps of the same color on the ends of the garland.

However, if you provide, say, 1 red, 10 green and 2 blue lamps then the store workers won't be able to build any garland of them. Any garland consisting of these lamps will have at least one pair of lamps of the same color adjacent to each other. Note that the store workers should use all the lamps you provided.

So Polycarp has bought some sets of lamps and now he wants to know if the store workers can build a garland from each of them.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of sets of lamps Polycarp has bought.

Each of the next t lines contains three integers r, g and b ($1 \le r, g, b \le 10^9$) — the number of red, green and blue lamps in the set, respectively.

Output

Print t lines — for each set of lamps print "Yes" if the store workers can build a garland from them and "No" otherwise.

Example

input
3 3 3 3 1 10 2 2 1 1
output
Yes No Yes

Note

The first two sets are desribed in the statement.

The third set produces garland "RBRG", for example.

B. Verse For Santa

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

New Year is coming! Vasya has prepared a New Year's verse and wants to recite it in front of Santa Claus.

Vasya's verse contains n parts. It takes a_i seconds to recite the i-th part. Vasya can't change the order of parts in the verse: firstly he recites the part which takes a_1 seconds, secondly — the part which takes a_2 seconds, and so on. After reciting the verse, Vasya will get the number of presents equal to the number of parts he fully recited.

Vasya can skip at most one part of the verse while reciting it (if he skips more than one part, then Santa will definitely notice it).

Santa will listen to Vasya's verse for no more than s seconds. For example, if $s = 10$, $a = [100, 9, 1, 1]$, and Vasya skips the first part of verse, then he gets two presents.

Note that it is possible to recite the whole verse (if there is enough time).

Determine which part Vasya needs to skip to obtain the maximum possible number of gifts. If Vasya shouldn't skip anything, print 0.

If there are multiple answers, print any of them.

You have to process t 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 two integers n and s ($1 \le n \le 10^5, 1 \le s \le 10^9$) — the number of parts in the verse and the maximum number of seconds Santa will listen to Vasya, respectively.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the time it takes to recite each part of the verse.

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

Output
For each test case print one integer — the number of the part that Vasya needs to skip to obtain the maximum number of gifts. If Vasya shouldn't skip any parts, print `0`.

Example

input
3 7 11 2 9 1 3 18 1 4 4 35 11 9 10 7 1 8 5
output
2 1 0

Note
In the first test case if Vasya skips the second part then he gets three gifts.

In the second test case no matter what part of the verse Vasya skips.

In the third test case Vasya can recite the whole verse.

C. Stack of Presents

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

Santa has to send presents to the kids. He has a large stack of n presents, numbered from 1 to n; the topmost present has number a_1, the next present is a_2, and so on; the bottom present has number a_n. All numbers are distinct.

Santa has a list of m **distinct** presents he has to send: b_1, b_2, ..., b_m. He will send them **in the order they appear in the list**.

To send a present, Santa has to find it in the stack by removing all presents above it, taking this present and returning all removed presents on top of the stack. So, if there are k presents above the present Santa wants to send, it takes him $2k + 1$ seconds to do it. Fortunately, Santa can speed the whole process up — when he returns the presents to the stack, he may reorder them as he wishes (only those which were above the present he wanted to take; the presents below cannot be affected in any way).

What is the minimum time required to send all of the presents, provided that Santa knows the whole list of presents he has to send and reorders the presents optimally? Santa cannot change the order of presents or interact with the stack of presents in any other way.

Your program has to answer t different test cases.

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

Then the test cases follow, each represented by three lines.

The first line contains two integers n and m ($1 \le m \le n \le 10^5$) — the number of presents in the stack and the number of presents Santa wants to send, respectively.

The second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le n$, all a_i are unique) — the order of presents in the stack.

The third line contains m integers b_1, b_2, ..., b_m ($1 \le b_i \le n$, all b_i are unique) — the ordered list of presents Santa has to send.

The sum of n over all test cases does not exceed 10^5.

Output

For each test case print one integer — the minimum number of seconds which Santa has to spend sending presents, if he reorders the presents optimally each time he returns them into the stack.

Example

input
2 3 3 3 1 2 3 2 1 7 2 2 1 7 3 4 5 6 3 1
output
5 8

D. Santa's Bot

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Santa Claus has received letters from n different kids throughout this year. Of course, each kid wants to get some presents from Santa: in particular, the i-th kid asked Santa to give them one of k_i different items as a present. Some items could have been asked by multiple kids.

Santa is really busy, so he wants the New Year Bot to choose the presents for all children. Unfortunately, the Bot's algorithm of choosing presents is bugged. To choose a present for some kid, the Bot does the following:

- choose one kid x equiprobably among all n kids;
- choose some item y equiprobably among all k_x items kid x wants;
- choose a kid z who will receive the present equipropably among all n kids (this choice is independent of choosing x and y); the resulting triple (x, y, z) is called **the decision** of the Bot.

If kid z listed item y as an item they want to receive, then the decision **valid**. Otherwise, the Bot's choice is **invalid**.

Santa is aware of the bug, but he can't estimate if this bug is really severe. To do so, he wants to know the probability that one decision generated according to the aforementioned algorithm is **valid**. Can you help him?

Input

The first line contains one integer n ($1 \le n \le 10^6$) — the number of kids who wrote their letters to Santa.

Then n lines follow, the i-th of them contains a list of items wanted by the i-th kid in the following format: k_i $a_{i,1}$ $a_{i,2}$ ... a_{i,k_i} ($1 \le k_i, a_{i,j} \le 10^6$), where k_i is the number of items wanted by the i-th kid, and $a_{i,j}$ are the items themselves. No item is contained in the same list more than once.

It is guaranteed that $\sum_{i=1}^{n} k_i \le 10^6$.

Output

Print the probatility that the Bot produces a **valid** decision as follows:

Let this probability be represented as an irreducible fraction $\frac{x}{y}$. You have to print $x \cdot y^{-1} \mod 998244353$, where y^{-1} is the inverse element of y modulo 998244353 (such integer that $y \cdot y^{-1}$ has remainder 1 modulo 998244353).

Examples

input
2 2 2 1 1 1
output
124780545

input
5 2 1 2 2 3 1 3 2 4 3 2 1 4 3 4 3 2
output
798595483

E. New Year Permutations

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

Yeah, we failed to make up a New Year legend for this problem.

A permutation of length n is an array of n integers such that every integer from 1 to n appears in it exactly once.

An element y of permutation p is reachable from element x if $x = y$, or $p_x = y$, or $p_{p_x} = y$, and so on.

The **decomposition** of a permutation p is defined as follows: firstly, we have a permutation p, all elements of which are **not marked**, and an empty list l. Then we do the following: while there is at least one **not marked** element in p, we find the leftmost such element, list all elements that are reachable from it **in the order they appear in** p, mark all of these elements, then cyclically shift the list of those elements so that the maximum appears at the first position, and add this list **as an element** of l. After all elements are marked, l is the result of this decomposition.

For example, if we want to build a decomposition of $p = [5, 4, 2, 3, 1, 7, 8, 6]$, we do the following:

1. initially $p = [5, 4, 2, 3, 1, 7, 8, 6]$ (bold elements are marked), $l = []$;
2. the leftmost unmarked element is 5; 5 and 1 are reachable from it, so the list we want to shift is $[5, 1]$; there is no need to shift it, since maximum is already the first element;
3. $p = [\mathbf{5}, 4, 2, 3, \mathbf{1}, 7, 8, 6]$, $l = [[5, 1]]$;
4. the leftmost unmarked element is 4, the list of reachable elements is $[4, 2, 3]$; the maximum is already the first element, so there's no need to shift it;
5. $p = [\mathbf{5}, \mathbf{4}, \mathbf{2}, \mathbf{3}, \mathbf{1}, 7, 8, 6]$, $l = [[5, 1], [4, 2, 3]]$;
6. the leftmost unmarked element is 7, the list of reachable elements is $[7, 8, 6]$; we have to shift it, so it becomes $[8, 6, 7]$;
7. $p = [\mathbf{5}, \mathbf{4}, \mathbf{2}, \mathbf{3}, \mathbf{1}, \mathbf{7}, \mathbf{8}, \mathbf{6}]$, $l = [[5, 1], [4, 2, 3], [8, 6, 7]]$;
8. all elements are marked, so $[[5, 1], [4, 2, 3], [8, 6, 7]]$ is the result.

The *New Year transformation* of a permutation is defined as follows: we build the decomposition of this permutation; then we sort all lists in decomposition in ascending order of the first elements (we don't swap the elements in these lists, only the lists themselves); then we concatenate the lists into one list which becomes a new permutation. For example, the *New Year transformation* of $p = [5, 4, 2, 3, 1, 7, 8, 6]$ is built as follows:

1. the decomposition is $[[5, 1], [4, 2, 3], [8, 6, 7]]$;
2. after sorting the decomposition, it becomes $[[4, 2, 3], [5, 1], [8, 6, 7]]$;
3. $[4, 2, 3, 5, 1, 8, 6, 7]$ is the result of the transformation.

We call a permutation **good** if the result of its transformation is the same as the permutation itself. For example, $[4, 3, 1, 2, 8, 5, 6, 7]$ is a good permutation; and $[5, 4, 2, 3, 1, 7, 8, 6]$ is bad, since the result of transformation is $[4, 2, 3, 5, 1, 8, 6, 7]$.

Your task is the following: given n and k, find the k-th (lexicographically) good permutation of length n.

Input

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

Then the test cases follow. Each test case is represented by one line containing two integers n and k ($1 \le n \le 50$, $1 \le k \le 10^{18}$).

Output

For each test case, print the answer to it as follows: if the number of good permutations of length n is less than k, print one integer -1; otherwise, print the k-th good permutation on n elements (in lexicographical order).

Example

input
5 3 3 5 15 4 13 6 8 4 2
output
2 1 3 3 1 2 5 4 -1 1 2 6 3 4 5 1 2 4 3

F. New Year and Handle Change

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

New Year is getting near. So it's time to change handles on codeforces. Mishka wants to change his handle but in such a way that people would not forget who he is.

To make it work, he only allowed to change letters case. More formally, during **one** handle change he can choose any segment of his handle $[i; i + l - 1]$ and apply `tolower` or `toupper` to all letters of his handle on this segment (more fomally, replace all uppercase letters with corresponding lowercase or vice versa). The length l is fixed for all changes.

Because it is not allowed to change codeforces handle too often, Mishka can perform at most k such operations. What is the **minimum** value of $min(lower, upper)$ (where $lower$ is the number of lowercase letters, and $upper$ is the number of uppercase letters) can be obtained after optimal sequence of changes?

Input

The first line of the input contains three integers n, k and l ($1 \le n, k, l \le 10^6, l \le n$) — the length of Mishka's handle, the number of changes and the length of the segment.

The second line of the input contains one string s, consisting of n lowercase and uppercase Latin letters — Mishka's handle.

Output

Print one integer — the minimum value of $min(lower, upper)$ after that Mishka change his handle at most k times in a way described in the problem statement.

Examples

input
7 1 4 PikMike
output
0

input
15 2 2 AaAaAAaaAAAAaaA
output
2

input
14 2 6 aBcdEFGHIJklMn
output
0

input
9 2 2 aAaAAAaaA
output
1