A. Skibidus and Amog'u

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

Skibidus lands on a foreign planet, where the local Amog tribe speaks the Amog'u language. In *Amog'u*, there are two forms of nouns, which are *singular* and *plural*.

Given that the root of the noun is transcribed as S, the two forms are transcribed as:

- Singular: $S + $ "us"
- Plural: $S + $ "i"

Here, $+$ denotes string concatenation. For example, $\texttt{abc} + \texttt{def} = \texttt{abcdef}$.

For example, when S is transcribed as "amog", then the singular form is transcribed as "amogus", and the plural form is transcribed as "amogi". Do note that *Amog'u* nouns can have an **empty** root — in specific, "us" is the singular form of "i" (which, on an unrelated note, means "imposter" and "imposters" respectively).

Given a transcribed *Amog'u* noun in singular form, please convert it to the transcription of the corresponding plural noun.

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 only line of each test case contains a string W, which is a transcribed *Amog'u* noun in *singular* form. It is guaranteed that W consists of only lowercase English letters, has a length of at most 10, and ends with "us".

Output

For each test case, output the transcription of the corresponding plural noun on a separate line.

Standard Input	Standard Output
9 us sus fungus cactus sussus amogus chungus ntarsus skibidus	i si fungi cacti sussi amogi chungi ntarsi skibidi

,

B. Skibidus and Ohio

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

Skibidus is given a string s that consists of lowercase Latin letters. If s contains more than 1 letter, he can:

- Choose an index i ($1 \leq i \leq |s| - 1$, $|s|$ denotes the current length of s) such that $s_i = s_{i+1}$. Replace s_i with any lowercase Latin letter of his choice. Remove s_{i+1} from the string.

Skibidus must determine the minimum possible length he can achieve through any number of operations.

Input

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

The only line of each test case contains a string s ($1 \leq |s| \leq 100$). It is guaranteed s only contains lowercase Latin letters.

Output

For each test case, output an integer on the new line, the minimum achievable length of s.

Standard Input	Standard Output
4 baa skibidus cc ohio	1 8 1 4

Note

In the first test case, Skibidus can:

- Perform an operation on $i = 2$. He replaces s_2 with b and removes s_3 from the string. Then, s becomes bb.
- Perform an operation on $i = 1$. He replaces s_1 with b and removes s_2 from the string. Then, s becomes b.
- Because s only contains 1 letter, Skibidus cannot perform any more operations.

Therefore, the answer is 1 for the first test case.

In the second test case, he cannot perform an operation on any index. Therefore, the answer is still the length of the initial string, 8.

,

C1. Skibidus and Fanum Tax (easy version)

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

This is the easy version of the problem. In this version, $m = 1$.

Skibidus has obtained two arrays a and b, containing n and m elements respectively. For **each** integer i from 1 to n, he is allowed to perform the operation **at most once**:

- Choose an integer j such that $1 \leq j \leq m$. Set $a_i := b_j - a_i$. Note that a_i may become non-positive as a result of this operation.

Skibidus needs your help determining whether he can sort a in non-decreasing order* by performing the above operation some number of times.

*a is sorted in non-decreasing order if $a_1 \leq a_2 \leq \ldots \leq a_n$.

Input

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

The first line of each test case contains two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $\mathbf{m = 1}$).

The following line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

The following line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 10^9$).

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

Output

For each test case, if it is possible to sort a in non-decreasing order, print "YES" on a new line. Otherwise, print "NO" on a new line.

You can output the answer in any case. For example, the strings "yEs", "yes", and "Yes" will also be recognized as positive responses.

Standard Input	Standard Output
5 1 1 5 9 3 1 1 4 3 3 4 1 1 4 2 5 6 4 1 5 4 10 5 4	YES NO YES NO YES

3 1 9 8 7 8	

Note

In the first test case, $[5]$ is already sorted.

In the second test case, it can be shown that it is impossible.

In the third test case, we can set $a_3 := b_1 - a_3 = 6 - 2 = 4$. The sequence $[1, 4, 4, 5]$ is in nondecreasing order.

In the last case, we can apply operations on each index. The sequence becomes $[-1, 0, 1]$, which is in nondecreasing order.

,

C2. Skibidus and Fanum Tax (hard version)

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

This is the hard version of the problem. In this version, $m \leq 2 \cdot 10^5$.

Skibidus has obtained two arrays a and b, containing n and m elements respectively. For **each** integer i from 1 to n, he is allowed to perform the operation **at most once**:

- Choose an integer j such that $1 \leq j \leq m$. Set $a_i := b_j - a_i$. Note that a_i may become non-positive as a result of this operation.

Skibidus needs your help determining whether he can sort a in non-decreasing order* by performing the above operation some number of times.

*a is sorted in non-decreasing order if $a_1 \leq a_2 \leq \ldots \leq a_n$.

Input

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

The first line of each test case contains two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq m \leq 2 \cdot 10^5$).

The following line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$).

The following line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \leq b_i \leq 10^9$).

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

Output

For each test case, if it is possible to sort a in non-decreasing order, print "YES" on a new line. Otherwise, print "NO" on a new line.

You can output the answer in any case. For example, the strings "yEs", "yes", and "Yes" will also be recognized as positive responses.

Standard Input	Standard Output
5 1 3 5 9 1 1000000000 3 2 1 4 3 3 4 4 3 2 4 6 5 6 1 8 5 2 6 4 5 4 5 4 1000	YES NO YES NO YES

3 1 9 8 7 8	

Note

In the first test case, $[5]$ is already sorted.

In the second test case, it can be shown that it is impossible.

In the third test case, we can set $a_2 := b_1 - a_2 = 6 - 4 = 2$ and $a_3 := b_3 - a_3 = 8 - 6 = 2$. The sequence $[2, 2, 2, 5]$ is in nondecreasing order.

In the last case, we can apply operations on each index. The sequence becomes $[-1, 0, 1]$, which is in nondecreasing order.

,

D. Skibidus and Sigma

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

Let's denote the score of an array b with k elements as $\sum_{i=1}^{k} \left(\sum_{j=1}^{i} b_j \right)$. In other words, let S_i denote the sum of the first i elements of b. Then, the score can be denoted as $S_1 + S_2 + \ldots + S_k$.

Skibidus is given n arrays $a_1, a_2, \ldots, a_n$, each of which contains m elements. Being the sigma that he is, he would like to concatenate them in **any order** to form a single array containing $n \cdot m$ elements. Please find the maximum possible score Skibidus can achieve with his concatenated array!

Formally, among all possible permutations* p of length n, output the maximum score of $a_{p_1} + a_{p_2} + \cdots + a_{p_n}$, where $+$ represents concatenation†.

*A permutation of length n contains all integers from 1 to n exactly once.

†The concatenation of two arrays c and d with lengths e and f respectively (i.e. $c + d$) is $c_1, c_2, \ldots, c_e, d_1, d_2, \ldots d_f$.

Input

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

The first line of each test case contains two integers n and m ($1 \leq n \cdot m \leq 2 \cdot 10^5$) — the number of arrays and the length of each array.

The i'th of the next n lines contains m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($1 \leq a_{i,j} \leq 10^6$) — the elements of the i'th array.

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

Output

For each test case, output the maximum score among all possible permutations p on a new line.

Standard Input	Standard Output
3 2 2 4 4 6 1 3 4 2 2 2 2 3 2 1 2 4 1 2 1 2 3 3 4 5 1 1 9	41 162 72

Note

For the first test case, there are two possibilities for p:

- $p = [1, 2]$. Then, $a_{p_1} + a_{p_2} = [4, 4, 6, 1]$. Its score is $4 + (4 + 4) + (4 + 4 + 6) + (4 + 4 + 6 + 1) = 41$.
- $p = [2, 1]$. Then, $a_{p_1} + a_{p_2} = [6, 1, 4, 4]$. Its score is $6 + (6 + 1) + (6 + 1 + 4) + (6 + 1 + 4 + 4) = 39$.

The maximum possible score is 41.

In the second test case, one optimal arrangement of the final concatenated array is $[4, 1, 2, 1, 2, 2, 2, 2, 3, 2, 1, 2]$. We can calculate that the score is 162.

,

E. Skibidus and Rizz

Input file: standard input
Output file: standard output
Time limit: 1.5 seconds
Memory limit: 256 megabytes

With the approach of Valentine's Day, Skibidus desperately needs a way to rizz up his crush! Fortunately, he knows of just the way: creating the perfect Binary String!

Given a binary string* t, let x represent the number of `0` in t and y represent the number of `1` in t. Its **balance-value** is defined as the value of $\max(x-y, y-x)$.

Skibidus gives you three integers n, m, and k. He asks for your help to construct a binary string s of length $n+m$ with exactly n `0`'s and m `1`'s such that the maximum **balance-value** among all of its substrings† is **exactly** k. If it is not possible, output `-1`.

---

*A binary string only consists of characters `0` and `1`.

†A string a is a substring of a string b if a can be obtained from b by the deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

Input

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

The first and only line of each test case contains three integers n, m, and k ($0 \leq n, m \leq 2 \cdot 10^5$, $1 \leq k \leq n+m$, $n+m \geq 1$).

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

Output

For each test case, if it is possible to construct s, output it on a new line. If there are multiple possible s, output any. Otherwise, output `-1` on a new line.

Standard Input	Standard Output
6 1 2 1 4 3 2 2 4 3 8 3 2 5 0 4 5 0 5	101 0100101 011011 -1 -1 00000

Note

In the first test case, we must construct s such that it contains one `0`, two `1`, and a maximum balance of 1 among all of its substrings. One possible valid s is `101` because:

- Consider the substring bounded by indices $[1, 1]$. Its **balance-value** is $\max(0-1, 1-0) = 1$.
- Consider the substring bounded by indices $[1, 2]$. Its **balance-value** is $\max(1-1, 1-1) = 0$.
- Consider the substring bounded by indices $[1, 3]$. Its **balance-value** is $\max(1-2, 2-1) = 1$.
- Consider the substring bounded by indices $[2, 2]$. Its **balance-value** is $\max(1-0, 0-1) = 1$.
- Consider the substring bounded by indices $[2, 3]$. Its **balance-value** is $\max(1-1, 1-1) = 0$.

- Consider the substring bounded by indices $[3, 3]$. Its **balance-value** is $\max(0 - 1, 1 - 0) = 1$.

Among all possible substrings, the maximum **balance-value** is 1.

In the second test case, the substring with the maximum **balance-value** is 0100, which has a balance of $max(3 - 1, 1 - 3) = 2$.

,

F. Skibidus and Slay

Input file: standard input
Output file: standard output
Time limit: 4 seconds
Memory limit: 512 megabytes

Let's define the *majority* of a sequence of k elements as the unique value that appears strictly more than $\left\lfloor \frac{k}{2} \right\rfloor$ times. If such a value does not exist, then the sequence does **not** have a majority. For example, the sequence $[1, 3, 2, 3, 3]$ has a majority 3 because it appears $3 > \left\lfloor \frac{5}{2} \right\rfloor = 2$ times, but $[1, 2, 3, 4, 5]$ and $[1, 3, 2, 3, 4]$ do not have a majority.

Skibidus found a tree* of n vertices and an array a of length n. Vertex i has the value a_i written on it, where a_i is an integer in the range $[1, n]$.

For each i from 1 to n, please determine if there exists a non-trivial simple path† such that i is the *majority* of the **sequence of integers written on the vertices** that form the path.

---

*A tree is a connected graph without cycles.

†A sequence of vertices $v_1, v_2, \ldots, v_m$ ($m \geq 2$) forms a non-trivial simple path if v_i and v_{i+1} are connected by an edge for all $1 \leq i \leq m-1$ and all v_i are pairwise distinct. **Note that the path must consist of at least 2 vertices.**

Input

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

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

The second line of each test case contains $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$) — the integers written on the vertices.

Each of the next $n-1$ lines contains two integers u_i and v_i, denoting the two vertices connected by an edge ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$).

It is guaranteed that the given edges form a tree.

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

Output

For each test case, output a binary string s of length n on a separate line. s_i should be computed as follows:

- If there is a non-trivial path containing i as the majority, s_i is '1';
- Otherwise, s_i is '0'.

Standard Input	Standard Output
4 3 1 2 3 1 3 2 3 4 3 1 1 3 1 2	000 1010 0001 1001001000100

2 3 4 2 4 2 4 4 2 1 2 2 3 3 4 13 1 4 4 7 4 7 1 1 7 11 11 11 11 1 2 2 3 3 4 4 5 4 6 2 7 7 8 2 9 6 10 5 11 11 12 10 13	

Note

In the first test case, there is no non-trivial path with 1, 2, or 3 as a majority, so the binary string outputted is "000".

In the second test case, $1 \rightarrow 2 \rightarrow 4$ is a non-trivial path with 3 as a majority.

,

G. Skibidus and Capping

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

Skibidus was abducted by aliens of Amog! Skibidus tries to talk his way out, but the Amog aliens don't believe him. To prove that he is not totally capping, the Amog aliens asked him to solve this task:

An integer x is considered a *semi-prime* if it can be written as $p \cdot q$ where p and q are (not necessarily distinct) prime numbers. For example, 9 is a *semi-prime* since it can be written as $3 \cdot 3$, and 3 is a prime number.

Skibidus was given an array a containing n integers. He must report the number of pairs (i, j) such that $i \leq j$ and $\operatorname{lcm}(a_i, a_j)$* is *semi-prime*.

*Given two integers x and y, $\operatorname{lcm}(x, y)$ denotes the least common multiple of x and y.

Input

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

The first line of each test case contains an integer n ($2 \leq n \leq 2 \cdot 10^5$).

The following line contains n integers $a_1, a_2, \ldots, a_n$ ($2 \leq a_i \leq n$).

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

Output

For each test case, output the number of ordered pairs of indices on a new line.

Standard Input	Standard Output
3 4 2 2 3 4 6 2 2 3 4 5 6 9 2 2 4 5 7 8 9 3 5	5 12 18

Note

In the first test case, the 5 pairs of indices are $(1, 3)$, $(1, 4)$, $(2, 3)$, $(2, 4)$, and $(4, 4)$.

,

H. Bro Thinks He's Him

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

Skibidus thinks he's Him! He proved it by solving this difficult task. Can you also prove yourself?

Given a binary string* t, $f(t)$ is defined as the minimum number of contiguous substrings, each consisting of identical characters, into which t can be partitioned. For example, $f(\texttt{00110001}) = 4$ because t can be partitioned as `[00][11][000][1]` where each bracketed segment consists of identical characters.

Skibidus gives you a binary string s and q queries. In each query, a single character of the string is flipped (i.e. `0` changes to `1` and `1` changes to `0`); changes are saved after the query is processed. After each query, output the sum over all $f(b)$ where b is a non-empty subsequence† of s, modulo $998\,244\,353$.

*A binary string consists of only characters `0` and `1`.

†A subsequence of a string is a string which can be obtained by removing several (possibly zero) characters from the original string.

Input

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

The first line of each test case contains a binary string s ($1 \leq |s| \leq 2 \cdot 10^5$).

The following line of each test case contains an integer q ($1 \leq q \leq 2 \cdot 10^5$) — the number of queries.

The following line contains q integers $v_1, v_2, \ldots, v_q$ ($1 \leq v_i \leq |s|$), denoting s_{v_i} is flipped for the i'th query.

It is guaranteed that the sum of $|s|$ and the sum of q over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output q integers on a single line — the answer after each query modulo $998\,244\,353$.

Standard Input	Standard Output
3 101 2 1 3 10110 3 1 2 3 101110101 5 7 2 4 4 1	10 7 61 59 67 1495 1169 1417 1169 1396

Note

In the first test case, s becomes `001` after the first query. Let's calculate the answer for each subsequence:

- $f(s_1) = f(\texttt{0}) = 1$
- $f(s_2) = f(\texttt{0}) = 1$

- $f(s_3) = f(\mathtt{1}) = 1$
- $f(s_1 s_2) = f(\mathtt{00}) = 1$
- $f(s_1 s_3) = f(\mathtt{01}) = 2$
- $f(s_2 s_3) = f(\mathtt{01}) = 2$
- $f(s_1 s_2 s_3) = f(\mathtt{001}) = 2$

The sum of these values is 10, modulo $998\,244\,353$.