Codeforces Round #697 (Div. 3)

A. Odd Divisor

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an integer n. Check if n has an **odd** divisor, greater than one (does there exist such a number x ($x > 1$) that n is divisible by x and x is odd).

For example, if $n = 6$, then there is $x = 3$. If $n = 4$, then such a number does not exist.

Input

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

Each test case contains one integer n ($2 \le n \le 10^{14}$).

Please note, that the input for some test cases won't fit into 32-bit integer type, so you should use at least 64-bit integer type in your programming language.

Output

For each test case, output on a separate line:

- "YES" if n has an **odd** divisor, greater than one;
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

Example

input
6 2 3 4 5 998244353 1099511627776
output
NO YES NO YES YES NO

B. New Year's Number

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp remembered the 2020-th year, and he is happy with the arrival of the new 2021-th year. To remember such a wonderful moment, Polycarp wants to represent the number n as the sum of a certain number of 2020 and a certain number of 2021.

For example, if:

- $n = 4041$, then the number n can be represented as the sum $2020 + 2021$;
- $n = 4042$, then the number n can be represented as the sum $2021 + 2021$;
- $n = 8081$, then the number n can be represented as the sum $2020 + 2020 + 2020 + 2021$;
- $n = 8079$, then the number n cannot be represented as the sum of the numbers 2020 and 2021.

Help Polycarp to find out whether the number n can be represented as the sum of a certain number of numbers 2020 and a certain number of numbers 2021.

Input

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

Each test case contains one integer n ($1 \le n \le 10^6$) — the number that Polycarp wants to represent as the sum of the numbers

2020 and 2021.

Output
For each test case, output on a separate line:

- "YES" if the number n is representable as the sum of a certain number of 2020 and a certain number of 2021;
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

Example

input
5 1 4041 4042 8081 8079
output
NO YES YES YES NO

C. Ball in Berland

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

At the school where Vasya is studying, preparations are underway for the graduation ceremony. One of the planned performances is a ball, which will be attended by pairs of boys and girls.

Each class must present two couples to the ball. In Vasya's class, a boys and b girls wish to participate. But not all boys and not all girls are ready to dance in pairs.

Formally, you know k possible one-boy-one-girl pairs. You need to choose two of these pairs so that no person is in more than one pair.

For example, if $a = 3$, $b = 4$, $k = 4$ and the couples $(1, 2)$, $(1, 3)$, $(2, 2)$, $(3, 4)$ are ready to dance together (in each pair, the boy's number comes first, then the girl's number), then the following combinations of two pairs are possible (not all possible options are listed below):

- $(1, 3)$ and $(2, 2)$;
- $(3, 4)$ and $(1, 3)$;

But the following combinations are not possible:

- $(1, 3)$ and $(1, 2)$ — the first boy enters two pairs;
- $(1, 2)$ and $(2, 2)$ — the second girl enters two pairs;

Find the number of ways to select two pairs that match the condition above. Two ways are considered different if they consist of different pairs.

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

The first line of each test case contains three integers a, b and k ($1 \le a, b, k \le 2 \cdot 10^5$) — the number of boys and girls in the class and the number of couples ready to dance together.

The second line of each test case contains k integers $a_1, a_2, \ldots a_k$. ($1 \le a_i \le a$), where a_i is the number of the boy in the pair with the number i.

The third line of each test case contains k integers $b_1, b_2, \ldots b_k$. ($1 \le b_i \le b$), where b_i is the number of the girl in the pair with the number i.

It is guaranteed that the sums of a, b, and k over all test cases do not exceed $2 \cdot 10^5$.

It is guaranteed that each pair is specified at most once in one test case.

Output
For each test case, on a separate line print one integer — the number of ways to choose two pairs that match the condition above.

Example

input

```
3
3 4 4
1 1 2 3
2 3 2 4
1 1 1
1
1
2 2 4
1 1 2 2
1 2 1 2
```

output

```
4
0
2
```

Note

In the first test case, the following combinations of pairs fit:

- $(1, 2)$ and $(3, 4)$;
- $(1, 3)$ and $(2, 2)$;
- $(1, 3)$ and $(3, 4)$;
- $(2, 2)$ and $(3, 4)$.

There is only one pair in the second test case.

In the third test case, the following combinations of pairs fit:

- $(1, 1)$ and $(2, 2)$;
- $(1, 2)$ and $(2, 1)$.

D. Cleaning the Phone

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

Polycarp often uses his smartphone. He has already installed n applications on it. Application with number i takes up a_i units of memory.

Polycarp wants to free at least m units of memory (by removing some applications).

Of course, some applications are more important to Polycarp than others. He came up with the following scoring system — he assigned an integer b_i to each application:

- $b_i = 1$ — regular application;
- $b_i = 2$ — important application.

According to this rating system, his phone has $b_1 + b_2 + \ldots + b_n$ convenience points.

Polycarp believes that if he removes applications with numbers $i_1, i_2, \ldots, i_k$, then he will free $a_{i_1} + a_{i_2} + \ldots + a_{i_k}$ units of memory and lose $b_{i_1} + b_{i_2} + \ldots + b_{i_k}$ convenience points.

For example, if $n = 5$, $m = 7$, $a = [5, 3, 2, 1, 4]$, $b = [2, 1, 1, 2, 1]$, then Polycarp can uninstall the following application sets (not all options are listed below):

- applications with numbers $1, 4$ and 5. In this case, it will free $a_1 + a_4 + a_5 = 10$ units of memory and lose $b_1 + b_4 + b_5 = 5$ convenience points;
- applications with numbers 1 and 3. In this case, it will free $a_1 + a_3 = 7$ units of memory and lose $b_1 + b_3 = 3$ convenience points.
- applications with numbers 2 and 5. In this case, it will free $a_2 + a_5 = 7$ memory units and lose $b_2 + b_5 = 2$ convenience points.

Help Polycarp, choose a set of applications, such that if removing them will free at least m units of memory and lose the minimum number of convenience points, or indicate that such a set does not exist.

Input

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

The first line of each test case contains two integers n and m ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le 10^9$) — the number of applications on Polycarp's phone and the number of memory units to be freed.

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 number of memory units used by applications.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($1 \le b_i \le 2$) — the convenience points of each application.

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 on a separate line:

- -1, if there is no set of applications, removing which will free at least m units of memory;
- the minimum number of convenience points that Polycarp will lose if such a set exists.

Example

input
5 5 7 5 3 2 1 4 2 1 1 2 1 1 3 2 1 5 10 2 3 2 3 2 1 2 1 2 1 4 10 5 1 3 4 1 2 1 2 4 5 3 2 1 2 2 1 2 1
output
2 -1 6 4 3

Note
In the first test case, it is optimal to remove applications with numbers 2 and 5, freeing 7 units of memory. $b_2 + b_5 = 2$.

In the second test case, by removing the only application, Polycarp will be able to clear only 2 of memory units out of the 3 needed.

In the third test case, it is optimal to remove applications with numbers 1, 2, 3 and 4, freeing 10 units of memory. $b_1 + b_2 + b_3 + b_4 = 6$.

In the fourth test case, it is optimal to remove applications with numbers 1, 3 and 4, freeing 12 units of memory. $b_1 + b_3 + b_4 = 4$.

In the fifth test case, it is optimal to remove applications with numbers 1 and 2, freeing 5 units of memory. $b_1 + b_2 = 3$.

E. Advertising Agency

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Masha works in an advertising agency. In order to promote the new brand, she wants to conclude contracts with some bloggers. In total, Masha has connections of n different bloggers. Blogger numbered i has a_i followers.

Since Masha has a limited budget, she can only sign a contract with k different bloggers. Of course, Masha wants her ad to be seen by as many people as possible. Therefore, she must hire bloggers with the maximum total number of followers.

Help her, find the number of ways to select k bloggers so that the total number of their followers is maximum possible. Two ways are considered different if there is at least one blogger in the first way, which is not in the second way. Masha believes that all bloggers have different followers (that is, there is no follower who would follow two different bloggers).

For example, if $n = 4$, $k = 3$, $a = [1, 3, 1, 2]$, then Masha has two ways to select 3 bloggers with the maximum total number of followers:

- conclude contracts with bloggers with numbers 1, 2 and 4. In this case, the number of followers will be equal to $a_1 + a_2 + a_4 = 6$.
- conclude contracts with bloggers with numbers 2, 3 and 4. In this case, the number of followers will be equal to $a_2 + a_3 + a_4 = 6$.

Since the answer can be quite large, **output it modulo** $10^9 + 7$.

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

The first line of each test case contains two integers n and k ($1 \le k \le n \le 1000$) — the number of bloggers and how many of them you can sign a contract with.

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

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

Output
For each test case, on a separate line output one integer — the number of ways to select k bloggers so that the total number of their followers is maximum possible.

Example

input
3 4 3 1 3 1 2 4 2 1 1 1 1 2 1 1 2
output
2 6 1

Note
The test case is explained in the statements.

In the second test case, the following ways are valid:

- conclude contracts with bloggers with numbers 1 and 2. In this case, the number of followers will be equal to $a_1 + a_2 = 2$;
- conclude contracts with bloggers with numbers 1 and 3. In this case, the number of followers will be equal to $a_1 + a_3 = 2$;
- conclude contracts with bloggers with numbers 1 and 4. In this case, the number of followers will be equal to $a_1 + a_4 = 2$;
- conclude contracts with bloggers with numbers 2 and 3. In this case, the number of followers will be equal to $a_2 + a_3 = 2$;
- conclude contracts with bloggers with numbers 2 and 4. In this case, the number of followers will be equal to $a_2 + a_4 = 2$;
- conclude contracts with bloggers with numbers 3 and 4. In this case, the number of followers will be equal to $a_3 + a_4 = 2$.

In the third test case, the following ways are valid:

- concludes a contract with a blogger with the number 2. In this case, the number of followers will be equal to $a_2 = 2$.

F. Unusual Matrix

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given two binary square matrices a and b of size $n \times n$. A matrix is called binary if each of its elements is equal to 0 or 1. You can do the following operations on the matrix a **arbitrary** number of times (0 or more):

- vertical `xor`. You choose the number j ($1 \le j \le n$) and for all i ($1 \le i \le n$) do the following: $a_{i,j} := a_{i,j} \oplus 1$ ($\oplus$ — is the operation xor (exclusive or)).
- horizontal `xor`. You choose the number i ($1 \le i \le n$) and for all j ($1 \le j \le n$) do the following: $a_{i,j} := a_{i,j} \oplus 1$.

Note that the elements of the a matrix change after each operation.

For example, if $n = 3$ and the matrix a is:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Then the following sequence of operations shows an example of transformations:

- vertical `xor`, $j = 1$.

$$a = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

- horizontal `xor`, $i = 2$.

$$a = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{pmatrix}$$

- vertical `xor`, $j = 2$.

$$a = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Check if there is a sequence of operations such that the matrix a becomes equal to the matrix b.

Input

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

The first line of each test case contains one integer n ($1 \le n \le 1000$) — the size of the matrices.

The following n lines contain strings of length n, consisting of the characters '0' and '1' — the description of the matrix a.

An empty line follows.

The following n lines contain strings of length n, consisting of the characters '0' and '1' — the description of the matrix b.

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

Output

For each test case, output on a separate line:

- "YES", there is such a sequence of operations that the matrix a becomes equal to the matrix b;
- "NO" otherwise.

You can output "YES" and "NO" in any case (for example, the strings yEs, yes, Yes and YES will be recognized as positive).

Example

input
3 3 110 001 110 000 000 000 3 101 010 101 010 101 010 2 01 11 10 10
output
YES YES NO

Note

The first test case is explained in the statements.

In the second test case, the following sequence of operations is suitable:

- horizontal xor, $i = 1$;
- horizontal xor, $i = 2$;
- horizontal xor, $i = 3$;

It can be proved that there is no sequence of operations in the third test case so that the matrix a becomes equal to the matrix b.

G. Strange Beauty

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

Polycarp found on the street an array a of n elements.

Polycarp invented his criterion for the beauty of an array. He calls an array a beautiful if at least one of the following conditions must be met **for each different pair of indices** $i \ne j$:

- a_i is divisible by a_j;
- or a_j is divisible by a_i.

For example, if:

- $n = 5$ and $a = [7, 9, 3, 14, 63]$, then the a array is not beautiful (for $i = 4$ and $j = 2$, none of the conditions above is met);
- $n = 3$ and $a = [2, 14, 42]$, then the a array is beautiful;
- $n = 4$ and $a = [45, 9, 3, 18]$, then the a array is not beautiful (for $i = 1$ and $j = 4$ none of the conditions above is met);

Ugly arrays upset Polycarp, so he wants to remove some elements from the array a so that it becomes beautiful. Help Polycarp determine the smallest number of elements to remove to make the array a beautiful.

Input

The first line contains one integer t ($1 \le t \le 10$) — the number of test cases. Then t test cases follow.

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

The second line of each test case contains n numbers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$) — elements of the array a.

Output

For each test case output one integer — the minimum number of elements that must be removed to make the array a beautiful.

Example

input
4 5 7 9 3 14 63 3 2 14 42 4 45 9 3 18 3 2 2 8
output
2 0 1 0

Note

In the first test case, removing 7 and 14 will make array a beautiful.

In the second test case, the array a is already beautiful.

In the third test case, removing one of the elements 45 or 18 will make the array a beautiful.

In the fourth test case, the array a is beautiful.