A. Least Product

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

You are given an array of integers $a_1, a_2, \ldots, a_n$. You can perform the following operation any number of times (possibly zero):

- Choose any element a_i from the array and change its value to any integer between 0 and a_i (inclusive). More formally, if $a_i < 0$, replace a_i with any integer in $[a_i, 0]$, otherwise replace a_i with any integer in $[0, a_i]$.

Let r be the minimum possible product of all the a_i after performing the operation any number of times.

Find the minimum number of operations required to make the product equal to r. Also, print one such shortest sequence of operations. If there are multiple answers, you can print any of them.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 500$) - the number of test cases. This is followed by their description.

The first line of each test case contains the a single integer n ($1 \le n \le 100$) — the length of the array.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($-10^9 \le a_i \le 10^9$).

Output

For each test case:

- The first line must contain the minimum number of operations k ($0 \le k \le n$).
- The j-th of the next k lines must contain two integers i and x, which represent the j-th operation. That operation consists in replacing a_i with x.

Standard Input	Standard Output
4 1 155 4 2 8 -1 3 4 -1 0 -2 -5 4 -15 -75 -25 -30	1 1 0 0 0 1 3 0

Note

In the first test case, we can change the value of the first integer into 0 and the product will become 0, which is the minimum possible.

In the second test case, initially, the product of integers is equal to $2 \cdot 8 \cdot (-1) \cdot 3 = -48$ which is the minimum possible, so we should do nothing in this case.

,

B. Erase First or Second Letter

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

You are given a string s of length n. Let's define two operations you can apply on the string:

- remove the first character of the string;
- remove the second character of the string.

Your task is to find the number of distinct **non-empty** strings that can be generated by applying the given operations on the initial string any number of times (possibly zero), in any order.

Input

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

The first line of each test case contains n ($1 \le n \le 10^5$) — the length of the string.

The second line of each test case contains the string s. It is guaranteed that the string only contains lowercase letters of the English alphabet.

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 a single integer: the number of distinct non-empty strings you can get.

Standard Input	Standard Output
5 5 aaaaa 1 z 5 ababa 14 bcdaaaabcdaaaa 20 abcdefghijklmnopqrst	5 1 9 50 210

Note

In the first test case, we can get the following strings: a, aa, aaa, $aaaa$, $aaaaa$.

In the third test case, for example, the word ba can be reached in the following way:

- remove the first character of the current string $ababa$, getting $baba$;
- remove the second character of the current string $baba$, getting bba;
- remove the second character of the current string bba, getting ba.

,

C. Watering an Array

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

You have an array of integers $a_1, a_2, \ldots, a_n$ of length n. On the i-th of the next d days you are going to do exactly one of the following two actions:

- Add 1 to each of the first b_i elements of the array a (i.e., set $a_j := a_j + 1$ for each $1 \le j \le b_i$).
- Count the elements which are equal to their position (i.e., the $a_j = j$). Denote the number of such elements as c. Then, you add c to your score, and reset the entire array a to a 0-array of length n (i.e., set $[a_1, a_2, \ldots, a_n] := [0, 0, \ldots, 0]$).

Your score is equal to 0 in the beginning. Note that on each day you should perform exactly one of the actions above: you cannot skip a day or perform both actions on the same day.

What is the maximum score you can achieve at the end?

Since d can be quite large, the sequence b is given to you in the compressed format:

- You are given a sequence of integers $v_1, v_2, \ldots, v_k$. The sequence b is a concatenation of infinitely many copies of v: $b = [v_1, v_2, \ldots, v_k, v_1, v_2, \ldots, v_k, \ldots]$.

Input

The first line contains a single integer t ($1 \le t \le 10^3$) — the number of test cases.

The first line of each test case contains three integers n, k and d ($1 \le n \le 2000$, $1 \le k \le 10^5$, $k \le d \le 10^9$) — the length of the array a, the length of the sequence v and the number of days you are going to perform operations on.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le n$) — the array a.

The third line of each test case contains k integers $v_1, v_2, \ldots, v_k$ ($1 \le v_i \le n$) — the sequence v.

It is guaranteed that the sum of n over all test cases doesn't exceed 2000 and the sum of k over all test cases doesn't exceed 10^5.

Output

For each test case, output one integer: the maximum score you can achieve at the end of the d-th day.

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

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

Note

In the first test case, the sequence b is equal to $[1, 3, 2, 3, 1, 3, 2, 3, \ldots]$ and one of the optimal solutions for this case is as follows:

- Perform the operation of the second type on the 1-st day: your score increases by 3 and array a becomes equal to $[0, 0, 0]$.
- Perform the operation of the first type on the 2-nd day: array a becomes equal to $[1, 1, 1]$.
- Perform the operation of the first type on the 3-rd day: array a becomes equal to $[2, 2, 1]$.
- Perform the operation of the second type on the 4-th day: your score increases by 1 and array a becomes equal to $[0, 0, 0]$.

It can be shown that it is impossible to score more than 4, so the answer is 4.

In the second test case, the sequence b is equal to $[6, 6, 6, 6, \ldots]$. One of the ways to score 3 is to perform operations of the first type on the 1-st and the 3-rd days and to perform an operation of the second type on the 2-nd day.

,

D. Yet Another Inversions Problem

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

You are given a permutation $p_0, p_1, \ldots, p_{n-1}$ of odd integers from 1 to $2n-1$ and a permutation $q_0, q_1, \ldots, q_{k-1}$ of integers from 0 to $k-1$.

An array $a_0, a_1, \ldots, a_{nk-1}$ of length nk is defined as follows:

$$a_{i \cdot k + j} = p_i \cdot 2^{q_j} \text{ for all } 0 \le i < n \text{ and all } 0 \le j < k$$

For example, if $p = [3, 5, 1]$ and $q = [0, 1]$, then $a = [3, 6, 5, 10, 1, 2]$.

Note that all arrays in the statement are zero-indexed. Note that each element of the array a is uniquely determined.

Find the number of inversions in the array a. Since this number can be very large, you should find only its remainder modulo $998\,244\,353$.

An inversion in array a is a pair (i, j) $(0 \le i < j < nk)$ such that $a_i > a_j$.

Input

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

The first line of each test case contains two integers n and k $(1 \le n, k \le 2 \cdot 10^5)$ — the lengths of arrays p and q.

The second line of each test case contains n distinct integers $p_0, p_1, \ldots, p_{n-1}$ $(1 \le p_i \le 2n-1, p_i$ is odd) — the array p.

The third line of each test case contains k distinct integers $q_0, q_1, \ldots, q_{k-1}$ $(0 \le q_i < k)$ — the array q.

It is guaranteed that the sum of n over all test cases doesn't exceed $2 \cdot 10^5$ and the sum of k over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, output one integer: the number of inversions in array a modulo $998\,244\,353$.

Standard Input	Standard Output
4 3 2 3 5 1 0 1 3 4 1 3 5 3 2 0 1 1 5 1 0 1 2 3 4 8 3	9 25 0 104

5 1 7 11 15 3 9 13 2 0 1	

Note

In the first test case, array a is equal to $[3, 6, 5, 10, 1, 2]$. There are 9 inversions in it: $(0, 4)$, $(0, 5)$, $(1, 2)$, $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, $(3, 4)$, $(3, 5)$. Note that these are pairs (i, j) such that $i < j$ and $a_i > a_j$.

In the second test case, array a is equal to $[8, 4, 1, 2, 24, 12, 3, 6, 40, 20, 5, 10]$. There are 25 inversions in it.

In the third test case, array a is equal to $[1, 2, 4, 8, 16]$. There are no inversions in it.

,

E. Construct Matrix

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

You are given an **even** integer n and an integer k. Your task is to construct a matrix of size $n \times n$ consisting of numbers 0 and 1 in such a way that the following conditions are true, or report that it is impossible:

- the sum of all the numbers in the matrix is exactly k;
- the bitwise `XOR` of all the numbers in the row i is the same for each i;
- the bitwise `XOR` of all the numbers in the column j is the same for each j.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 130$) — the number of test cases. The description of the test cases follows.

Each test case is described by a single line, which contains two integers n and k ($2 \le n \le 1000$, n is even, $0 \le k \le n^2$).

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, output `Yes` if it's possible to construct a matrix that satisfies all of the problem's conditions, and `No` otherwise.

If it is possible to construct a matrix, the i-th of the next n lines should contain n integers representing the elements in the i-th row of the matrix.

Standard Input	Standard Output
5 4 0 6 6 6 5 4 2 6 36	Yes 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 Yes 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 1 No No Yes 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

	1 1 1 1 1 1 1 1 1 1 1 1

Note

In the first example, all conditions are satisfied:

- the sum of all the numbers in the matrix is exactly 0;
- the bitwise `XOR` of all the numbers in the row i is 0 for each i;
- the bitwise `XOR` of all the numbers in the column j is 0 for each j.

In the third example, it can be shown that it's impossible to find a matrix satisfying all the problem's conditions.

,

F. Construct Tree

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

You are given an array of integers $l_1, l_2, \ldots, l_n$ and an integer d. Is it possible to construct a tree satisfying the following three conditions?

- The tree contains $n + 1$ nodes.
- The length of the i-th edge is equal to l_i.
- The (weighted) diameter of the tree is equal to d.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 250$) — the number of test cases. The description of the test cases follows.

The first line of each test case contains two integers n, d ($2 \le n \le 2000, 1 \le d \le 2000$).

The second line of each test case contains n integers $l_1, l_2, \ldots, l_n$ ($1 \le l_i \le d$).

It is guaranteed that the sum of n over all test cases does not exceed 2000.

Output

For each test case, output `Yes` if it is possible to construct a tree that satisfies all the conditions, and `No` otherwise.

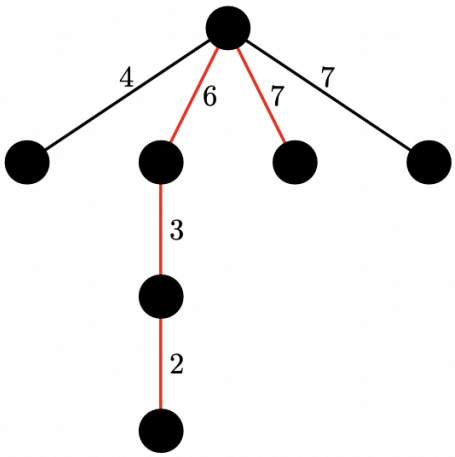
You can print the letters in any case (upper or lower).

Standard Input	Standard Output
3 4 10 1 2 3 4 4 7 1 4 3 4 6 18 2 4 3 7 6 7	Yes No Yes

Note

Below, you are given the illustrations of trees for the first and third test cases. One of the diameters is highlighted by coloring its edges in red.


1
2
3
4


4
6
7
7
3
2