A. Permutation Counting

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

You have some cards. An integer between 1 and n is written on each card: specifically, for each i from 1 to n, you have a_i cards which have the number i written on them.

There is also a shop which contains unlimited cards of each type. You have k coins, so you can buy k new cards in total, and the cards you buy can contain any integer between 1 and n.

After buying the new cards, you rearrange all your cards in a line. The score of a rearrangement is the number of (contiguous) subarrays of length n which are a permutation of $[1, 2, \ldots, n]$. What's the maximum score you can get?

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 first line of each test case contains two integers n, k ($1 \le n \le 2 \cdot 10^5$, $0 \le k \le 10^{12}$) — the number of distinct types of cards and the number of coins.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le 10^{12})$ — the number of cards of type i you have at the beginning.

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 single line containing an integer: the maximum score you can get.

Standard Input	Standard Output
8 1 10 1 2 4 8 4 3 4 6 1 8 3 9 7 6 2 5 3 6 6 7 4 6 9 7 7 6 1 7 6 2 4 3 3 10 10 1 3 1 2 1 9 3 5 7 5 9 8 5 8 7 5 1 3 2 9 8	11 15 15 22 28 32 28 36

Note

In the first test case, the final (and only) array we can get is $[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]$ (including 11 single 1s), which contains 11 subarrays consisting of a permutation of $[1]$.

In the second test case, we can buy 0 cards of type 1 and 4 cards of type 2, and then we rearrange the cards as following: $[1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2]$. There are 8 subarrays equal to $[1, 2]$ and 7 subarrays equal to $[2, 1]$, which make a total of 15 subarrays which are a permutation of $[1, 2]$. It can also be proved that this is the maximum score we can get.

In the third test case, one of the possible optimal rearrangements is $[3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 3]$.

,

B1. Reverse Card (Easy Version)

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

The two versions are different problems. You may want to read both versions. You can make hacks only if both versions are solved.

You are given two positive integers n, m.

Calculate the number of ordered pairs (a, b) satisfying the following conditions:

- $1 \le a \le n, 1 \le b \le m$;
- $a + b$ is a multiple of $b \cdot \gcd(a, b)$.

Input

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

The first line of each test case contains two integers n, m ($1 \le n, m \le 2 \cdot 10^6$).

It is guaranteed that neither the sum of n nor the sum of m over all test cases exceeds $2 \cdot 10^6$.

Output

For each test case, print a single integer: the number of valid pairs.

Standard Input	Standard Output
6 1 1 2 3 3 5 10 8 100 1233 1000000 1145141	1 3 4 14 153 1643498

Note

In the first test case, only $(1, 1)$ satisfies the conditions.

In the fourth test case, $(1, 1), (2, 1), (2, 2), (3, 1), (4, 1), (5, 1), (6, 1), (6, 2), (6, 3), (7, 1), (8, 1), (9, 1), (10, 1), (10, 2)$ satisfy the conditions.

,

B2. Reverse Card (Hard Version)

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

The two versions are different problems. You may want to read both versions. You can make hacks only if both versions are solved.

You are given two positive integers n, m.

Calculate the number of ordered pairs (a, b) satisfying the following conditions:

- $1 \le a \le n$, $1 \le b \le m$;
- $b \cdot \gcd(a, b)$ is a multiple of $a + b$.

Input

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

The first line of each test case contains two integers n, m ($1 \le n, m \le 2 \cdot 10^6$).

It is guaranteed that neither the sum of n nor the sum of m over all test cases exceeds $2 \cdot 10^6$.

Output

For each test case, print a single integer: the number of valid pairs.

Standard Input	Standard Output
6 1 1 2 3 3 5 10 8 100 1233 1000000 1145141	0 1 1 6 423 5933961

Note

In the first test case, no pair satisfies the conditions.

In the fourth test case, $(2, 2), (3, 6), (4, 4), (6, 3), (6, 6), (8, 8)$ satisfy the conditions.

,

C. Fenwick Tree

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

Let $\operatorname{lowbit}(x)$ denote the value of the lowest binary bit of x, e.g. $\operatorname{lowbit}(12) = 4$, $\operatorname{lowbit}(8) = 8$.

For an array a of length n, if an array s of length n satisfies $s_k = \left(\sum\limits_{i=k-\operatorname{lowbit}(k)+1}^{k} a_i\right) \bmod 998\,244\,353$ for all k, then s is called the *Fenwick Tree* of a. Let's denote it as $s = f(a)$.

For a positive integer k and an array a, $f^k(a)$ is defined as follows:

$$f^k(a) = \begin{cases} f(a) & \text{if } k = 1 \\ f(f^{k-1}(a)) & \text{otherwise.} \end{cases}$$

You are given an array b of length n and a positive integer k. Find an array a that satisfies $0 \le a_i < 998\,244\,353$ and $f^k(a) = b$. It can be proved that an answer always exists. If there are multiple possible answers, you may print any of them.

Input

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

The first line of each test case contains two positive integers n ($1 \le n \le 2 \cdot 10^5$) and k ($1 \le k \le 10^9$), representing the length of the array and the number of times the function f is performed.

The second line of each test case contains an array $b_1, b_2, \ldots, b_n$ ($0 \le b_i < 998\,244\,353$).

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

Output

For each test case, print a single line, containing a valid array a of length n.

Standard Input	Standard Output
2 8 1 1 2 1 4 1 2 1 8 6 2 1 4 3 17 5 16	1 1 1 1 1 1 1 1 1 2 3 4 5 6

Note

In the first test case, it can be seen that $f^1([1,1,1,1,1,1,1,1]) = [1,2,1,4,1,2,1,8]$.

In the second test case, it can be seen that $f^2([1,2,3,4,5,6]) = f^1([1,3,3,10,5,11]) = [1,4,3,17,5,16]$.

,

D. Long Way to be Non-decreasing

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

Little R is a magician who likes non-decreasing arrays. She has an array of length n, initially as $a_1, \ldots, a_n$, in which each element is an integer between $[1, m]$. She wants it to be non-decreasing, i.e., $a_1 \le a_2 \le \ldots \le a_n$.

To do this, she can perform several magic tricks. Little R has a fixed array $b_1 \ldots b_m$ of length m. Formally, let's define a trick as a procedure that does the following things in order:

- Choose a set $S \subseteq \{1, 2, \ldots, n\}$.
- For each $u \in S$, assign a_u with b_{a_u}.

Little R wonders how many tricks are needed at least to make the initial array non-decreasing. If it is not possible with any amount of tricks, print -1 instead.

Input

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

The first line of each test case contains two integers n and m ($1 \le n \le 10^6$, $1 \le m \le 10^6$) — the length of the initial array and the range of the elements in the array.

The second line of each test case contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le m$) — the initial array.

The third line of each test case contains m integers $b_1, \ldots, b_m$ ($1 \le b_i \le m$) — the fixed magic array.

It is guaranteed that the sum of n over all test cases does not exceed 10^6 and the sum of m over all test cases does not exceed 10^6.

Output

For each test case, output a single integer: the minimum number of tricks needed, or -1 if it is impossible to make $a_1, \ldots, a_n$ non-decreasing.

Standard Input	Standard Output
3 5 8 1 6 3 7 1 2 3 5 8 7 1 5 6 3 3 1 3 2 2 1 3 10 10 2 8 5 4 8 4 1 5 10 10 6 7 2 6 3 4 1 1 3 5	3 -1 3

Note

In the first case, the initial array $a_1, \ldots, a_n$ is $[1, 6, 3, 7, 1]$. You can choose S as follows:

- first trick: $S = [2, 4, 5]$, $a = [1, 1, 3, 5, 2]$;
- second trick: $S = [5]$, $a = [1, 1, 3, 5, 3]$;
- third trick: $S = [5]$, $a = [1, 1, 3, 5, 5]$.

So it is possible to make $a_1, \ldots, a_n$ non-decreasing using 3 tricks. It can be shown that this is the minimum possible amount of tricks.

In the second case, it is impossible to make $a_1, \ldots, a_n$ non-decreasing.

,

E1. Again Counting Arrays (Easy Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     5 seconds
Memory limit:   512 megabytes
```

This is the easy version of the problem. The differences between the two versions are the constraints on n, m, b_0 and the time limit. You can make hacks only if both versions are solved.

Little R has counted many sets before, and now she decides to count arrays.

Little R thinks an array $b_0, \ldots, b_n$ consisting of non-negative integers is *continuous* if and only if, for each i such that $1 \le i \le n$, $|b_i - b_{i-1}| = 1$ is satisfied. She likes continuity, so she only wants to generate continuous arrays.

If Little R is given b_0 and $a_1, \ldots, a_n$, she will try to generate a non-negative continuous array b, which has no similarity with a. More formally, for all $1 \le i \le n$, $a_i \ne b_i$ holds.

However, Little R does not have any array a. Instead, she gives you n, m and b_0. She wants to count the different integer arrays $a_1, \ldots, a_n$ satisfying:

- $1 \le a_i \le m$;
- At least one non-negative continuous array $b_0, \ldots, b_n$ can be generated.

Note that $b_i \ge 0$, but the b_i can be arbitrarily large.

Since the actual answer may be enormous, please just tell her the answer modulo $998\,244\,353$.

Input

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

The first and only line of each test case contains three integers n, m, and b_0 $(1 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$, $0 \le b_0 \le 2 \cdot 10^5)$ — the length of the array $a_1, \ldots, a_n$, the maximum possible element in $a_1, \ldots, a_n$, and the initial element of the array $b_0, \ldots, b_n$.

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

Output

For each test case, output a single line containing an integer: the number of different arrays $a_1, \ldots, a_n$ satisfying the conditions, modulo $998\,244\,353$.

Standard Input	Standard Output
6 3 2 1 5 5 3 13 4 1 100 6 7 100 11 3 1000 424 132	6 3120 59982228 943484039 644081522 501350342

Note

In the first test case, for example, when $a = [1, 2, 1]$, we can set $b = [1, 0, 1, 0]$. When $a = [1, 1, 2]$, we can set $b = [1, 2, 3, 4]$. In total, there are 6 valid choices of $a_1, \ldots, a_n$: in fact, it can be proved that only $a = [2, 1, 1]$ and $a = [2, 1, 2]$ make it impossible to construct a non-negative continuous $b_0, \ldots, b_n$, so the answer is $2^3 - 2 = 6$.

,

E2. Again Counting Arrays (Hard Version)

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   512 megabytes
```

This is the hard version of the problem. The differences between the two versions are the constraints on n, m, b_0 and the time limit. You can make hacks only if both versions are solved.

Little R has counted many sets before, and now she decides to count arrays.

Little R thinks an array $b_0, \ldots, b_n$ consisting of non-negative integers is *continuous* if and only if, for each i such that $1 \le i \le n$, $|b_i - b_{i-1}| = 1$ is satisfied. She likes continuity, so she only wants to generate continuous arrays.

If Little R is given b_0 and $a_1, \ldots, a_n$, she will try to generate a non-negative continuous array b, which has no similarity with a. More formally, for all $1 \le i \le n$, $a_i \ne b_i$ holds.

However, Little R does not have any array a. Instead, she gives you n, m and b_0. She wants to count the different integer arrays $a_1, \ldots, a_n$ satisfying:

- $1 \le a_i \le m$;
- At least one non-negative continuous array $b_0, \ldots, b_n$ can be generated.

Note that $b_i \ge 0$, but the b_i can be arbitrarily large.

Since the actual answer may be enormous, please just tell her the answer modulo $998\,244\,353$.

Input

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

The first and only line of each test case contains three integers n, m, and b_0 $(1 \le n \le 2 \cdot 10^6$, $1 \le m \le 2 \cdot 10^6$, $0 \le b_0 \le 2 \cdot 10^6)$ — the length of the array $a_1, \ldots, a_n$, the maximum possible element in $a_1, \ldots, a_n$, and the initial element of the array $b_0, \ldots, b_n$.

It is guaranteed that the sum of n over all test cases does not exceeds 10^7.

Output

For each test case, output a single line containing an integer: the number of different arrays $a_1, \ldots, a_n$ satisfying the conditions, modulo $998\,244\,353$.

Standard Input	Standard Output
6 3 2 1 5 5 3 13 4 1 100 6 7 100 11 3 1000 424 132	6 3120 59982228 943484039 644081522 501350342

Note

In the first test case, for example, when $a = [1, 2, 1]$, we can set $b = [1, 0, 1, 0]$. When $a = [1, 1, 2]$, we can set $b = [1, 2, 3, 4]$. In total, there are 6 valid choices of $a_1, \ldots, a_n$: in fact, it can be proved that only $a = [2, 1, 1]$ and $a = [2, 1, 2]$ make it impossible to construct a non-negative continuous $b_0, \ldots, b_n$, so the answer is $2^3 - 2 = 6$.

F. Next and Prev

```
Input file:     standard input
Output file:    standard output
Time limit:     15 seconds
Memory limit:   1024 megabytes
```

Let $p_1, \ldots, p_n$ be a permutation of $[1, \ldots, n]$.

Let the q-subsequence of p be a permutation of $[1, q]$, whose elements are in the same relative order as in $p_1, \ldots, p_n$. That is, we extract all elements not exceeding q together from p in the original order, and they make the q-subsequence of p.

For a given array a, let $pre(i)$ be the largest value satisfying $pre(i) < i$ and $a_{pre(i)} > a_i$. If it does not exist, let $pre(i) = -10^{100}$. Let $nxt(i)$ be the smallest value satisfying $nxt(i) > i$ and $a_{nxt(i)} > a_i$. If it does not exist, let $nxt(i) = 10^{100}$.

For each q such that $1 \leq q \leq n$, let $a_1, \ldots, a_q$ be the q-subsequence of p. For each i such that $1 \leq i \leq q$, $pre(i)$ and $nxt(i)$ will be calculated as defined. Then, you will be given some integer values of x, and for each of them you have to calculate $\sum_{i=1}^{q} \min(nxt(i) - pre(i), x)$.

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 ($1 \leq n \leq 3 \cdot 10^5$) — the length of the permutation.

The second line of each test case contains n integers $p_1, \ldots, p_n$ ($1 \leq p_i \leq n$) — the initial permutation.

Then, for each q such that $1 \leq q \leq n$ in ascending order, you will be given an integer k ($0 \leq k \leq 10^5$), representing the number of queries for the q-subsequence. Then k numbers in a line follow: each of them is the value of x for a single query ($1 \leq x \leq q$).

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$ and the sum of k over all test cases does not exceed 10^5.

Output

For each test case, for each query, print a single line with an integer: the answer to the query.

Standard Input	Standard Output
1 7 6 1 4 3 2 5 7 1 1 0 1 3 1 2 3 1 2 3 1 3 2 2 6	1 9 8 5 10 14 16 14 30

Note

The 1-subsequence is $[1]$, and $pre=[-10^{100}]$, $nxt=[10^{100}]$.
$ans(1)=\min(10^{100}-(-10^{100}),1)=1$.

The 5-subsequence is $[1,4,3,2,5]$, and $pre=[-10^{100},-10^{100},2,3,-10^{100}]$, $nxt=[2,5,5,5,10^{100}]$.
$ans(1)=5, ans(2)=10, ans(3)=14$.