A. Contest Proposal

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

A contest contains n problems and the difficulty of the i-th problem is expected to be **at most** b_i. There are already n problem proposals and the difficulty of the i-th problem is a_i. Initially, both $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$ are sorted in non-decreasing order.

Some of the problems may be more difficult than expected, so the writers must propose more problems. When a new problem with difficulty w is proposed, the most difficult problem will be deleted from the contest, and the problems will be sorted in a way that the difficulties are non-decreasing.

In other words, in each operation, you choose an integer w, insert it into the array a, sort array a in non-decreasing order, and remove the last element from it.

Find the minimum number of new problems to make $a_i \le b_i$ for all i.

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 only one positive integer n ($1 \le n \le 100$), representing the number of problems.

The second line of each test case contains an array a of length n ($1 \le a_1 \le a_2 \le \cdots \le a_n \le 10^9$).

The third line of each test case contains an array b of length n ($1 \le b_1 \le b_2 \le \cdots \le b_n \le 10^9$).

Output

For each test case, print an integer as your answer in a new line.

Standard Input	Standard Output

2
6
1000 1400 2000 2000 2200 2700
800 1200 1500 1800 2200 3000
6
4 5 6 7 8 9
1 2 3 4 5 6
	2
	3

Note

In the first test case:

- Propose a problem with difficulty $w = 800$ and a becomes $[800, 1000, 1400, 2000, 2000, 2200]$.
- Propose a problem with difficulty $w = 1800$ and a becomes $[800, 1000, 1400, 1800, 2000, 2000]$.

It can be proved that it's impossible to reach the goal by proposing fewer new problems.

In the second test case:

- Propose a problem with difficulty $w = 1$ and a becomes $[1, 4, 5, 6, 7, 8]$.
- Propose a problem with difficulty $w = 2$ and a becomes $[1, 2, 4, 5, 6, 7]$.
- Propose a problem with difficulty $w = 3$ and a becomes $[1, 2, 3, 4, 5, 6]$.

It can be proved that it's impossible to reach the goal by proposing fewer new problems.

,

B. Coin Games

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

There are n coins on the table forming a circle, and each coin is either facing up or facing down. Alice and Bob take turns to play the following game, and Alice goes first.

In each operation, the player chooses a facing-up coin, removes the coin, and flips the two coins that are adjacent to it. If (before the operation) there are only two coins left, then one will be removed and the other won't be flipped (as it would be flipped twice). If (before the operation) there is only one coin left, no coins will be flipped. If (before the operation) there are no facing-up coins, the player loses.

Decide who will win the game if they both play optimally. It can be proved that the game will end in a finite number of operations, and one of them will win.

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 only one positive integer n $(1 \le n \le 100)$, representing the number of the coins.

A string s of length n follows on the second line of each test case, containing only "U" and "D", representing that each coin is facing up or facing down.

Output

For each test case, print "YES" if Alice will win the game, and "NO" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output
3 5 UUDUD 5 UDDUD 2 UU	YES NO NO

Note

In the first test case, the game may go as follows.

- Alice chooses the first coin and s becomes "DDUU".
- Bob chooses the last coin and s becomes "UDD".
- Alice chooses the first coin and s becomes "UU".
- Bob chooses the first coin and s becomes "U".
- Alice chooses the only coin and s becomes empty.

- Bob can't choose any coin now, and he loses the game.

It can be proved that Bob will always lose if they both play optimally.

C. 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	11
1 10	15
1	15
2 4	22
8 4	28
3 4	32
6 1 8	28
3 9	36
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	

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 $[3, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 2, 3, 1, 3]$.

,

D1. 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.

,

D2. 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.

,

E. Fenwick Tree

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

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

For an array a of length n, if an array s of length n satisfies $s_k = \left(\sum_{i=k-\mathrm{lowbit}(k)+1}^{k} a_i\right) \bmod 998\,244\,353$ 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 $f^2([1,2,3,4,5,6]) = f^1([1,3,3,10,5,11]) = [1,4,3,17,5,16]$.

,

F. 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 $a_1, \dots, a_n$ 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 $a_1, \dots, a_n$ 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 $a_1, \dots, a_n$ non-decreasing.