Official Problem Set

2017 ACM ICPC

ASIA, INDIA REGIONAL

ONLINE CONTEST

Problem code: **EQUALMOD**
Problem name: **Modulo Equality**

You have two arrays **A** and **B**, each containing **N** integers.

Elements of array **B** cannot be modified.

You may perform an arbitrary number of operations (including zero number of operations). In one operation, you can choose one index i ($1 \le i \le N$) and increment A_i by 1. Each index can be chosen any number of times.

Find the minimum number of operations required to modify the array **A** such that for every pair i, j ($1 \le i, j \le N$), ($A_i \% B_i$) equals ($A_j \% B_j$). Here, % denotes the modulo operator (the remainder after dividing by B_i). Note that $0 \le (A_i \% B_i) < B_i$.

Input

First line of the input contains an integer **T** denoting the number of testcases. The description of **T** testcases is as follows:

First line of each testcase contains a positive integer **N**, denoting the number of elements in arrays **A** and **B**.

Second line contains **N** space-separated whole numbers $A_1, A_2, ..., A_N$.

Third line contains **N** space-separated natural numbers $B_1, B_2, ..., B_N$.

Output

For each testcase, output a single line containing the minimum number of operations required to modify the array **A** in order to satisfy the required conditions.

Constraints

- $1 \le T \le 1000$
- $1 \le N \le 5 * 10^5$
- $0 \le A_i \le 10^9$
- $1 \le B_i \le 10^9$
- Sum of **N** across all testcases in a single file will be $\le 5 * 10^5$

Example

```
Input:
2
3
4 2 2
5 3 4
4
2 1 1 2
3 3 3 3

Output:
3
2
```

Explanation

Example case 1. A_1 can be increased from **4** to **7** in **3 operations** so that:

- $A_1 \% B_1 = 2$
- $A_2 \% B_2 = 2$
- $A_3 \% B_3 = 2$

Example case 2. A_2, A_3 can **both be increased** from **1** to **2** in **2 operations** total so that:

- $A_1 \% B_1 = 2$
- $A_2 \% B_2 = 2$
- $A_3 \% B_3 = 2$
- $A_4 \% B_4 = 2$

Problem code: **STDDEV**
Problem name: **Obtain Desired Standard Deviation**

Given two integers **N** and **σ**, find any array **a** with length **N** such that the standard deviation of the elements of the array is equal to **σ**. The standard deviation is given by the following formula:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (a_i - \mu)^2}{N}}, \text{where } \mu = \frac{\sum_{i=1}^{N} a_i}{N}$$

For each $\mathbf{1 \le i \le N}$, $\mathbf{a_i}$ may be any real number with $\mathbf{|a_i| \le 10^8}$.

If there is more than one possible answer, you may output any one of them. If there is no such array, output -1 instead.

Please note that the denominator only contains **N**. (In the usual formula for standard deviation, **N-1** is used. That's not the definition we're using here.)

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of the test cases follows.

The first and only line of each test case contains two space separated integers **N, σ**.

Output

For each test case, output a single line containing **N** space-separated real numbers denoting the elements of the array **a**.

If it is not possible to construct such an array, output -1 instead. If there is more than one possible solution, you may output any one.

Your answer will be considered correct if the standard deviation of the printed array doesn't differ from σ by more than 10^{-2} in absolute value.

Constraints

- **$1 \leq T \leq 10^5$**
- **$1 \leq N \leq 10^5$**
- **$0 \leq \sigma \leq 10^5$**
- Sum of **N** over all test cases won't exceed **10^6**

Example

Input

```
3
2 2
3 3
4 0
```

Output

```
2 6
3.5 2 8.979967
3 3 3 3
```

Problem code: **ORDTEAMS**
Problem name: **Ordering teams**

In ACM-ICPC contests, there are usually three people in a team. For each person in the team, you know their scores in three skills - hard work, intelligence and persistence.

You want to check whether it is possible to order these people (assign them numbers from 1 to 3) in such a way that for each $1 \le i \le 2$, $i+1$-th person is stricly *better* than the i-th person.

A person **x** is said to be *better* than another person **y** if **x** doesn't score less than **y** in any of the skills and scores more than **y** in at least one skill.

Determine whether such an ordering exists.

Input

The first line fo the input contains an integer **T** denoting the number of test cases.

Each test consists of three lines. Each of these lines contains three space separated integers s_1, s_2 and s_3 denoting the scores of one member of the team in each of the three skills, in the given order.

Output

For each test case, output a single line containing "yes" if such an ordering exists or "no" if doesn't exist (without quotes).

Constraints

- $1 \le T \le 1000$
- $1 \le s_1, s_2, s_3 \le 100$

Example

Input
```
3
1 2 3
2 3 4
2 3 5
1 2 3
2 3 4
2 3 4
5 6 5
1 2 3
2 3 4
```

Output
```
yes
no
yes
```

Explanation

Test Case 1: We can order them as (3, 2, 1). Person 3 is *better* than Person 2 because his scores in the first two skills are not lesser than Person 2's. And in skill 3, Person 3 scores higher. Similarly, Person 2 is *better* than Person 1. He scores more than Person 1 in every skill, in fact.

Problem code: **TWOTREES**
Problem name: **A tale of two trees**

There are two trees; tree 1 has $\mathbf{N_1}$ nodes which are numbered from 1 to $\mathbf{N_1}$, and tree 2 has $\mathbf{N_2}$ nodes which are numbered from 1 to $\mathbf{N_2}$. Each tree is rooted at its node 1.

You should perform the following operation: choose an arbitrary node **u** in tree 2, remove the subtree of node **u** (the subtree formed by node **u** and all its direct or indirect descendants) from tree 2 and add the subtree to any leaf of tree 1 (ie. **u** should become a child of some leaf in tree 1). Note that **u** can also be the root of tree 2. Afterwards, discard the rest of tree 2 and keep only the modified tree 1. How many distinct trees can you obtain by performing this operation exactly once?

Note that tree 1 will remain rooted in its own node 1. Two trees are considered identical if you can reorder (without changing the root) the children in each node and renumber the nodes (except the root) to make them equal. (Intuitively, the structure of the trees must be same - we're dealing with rooted, unlabelled, unordered isomorphic trees.)

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The first line of each test case contains two space-separated integers $\mathbf{N_1}$ and $\mathbf{N_2}$ denoting the number of nodes in tree 1 and tree 2 respectively.

The second line contains $\mathbf{N_1 - 1}$ space-separated integers $\mathbf{A_2}$, $\mathbf{A_3}$, ..., $\mathbf{A_{N1}}$ with each $\mathbf{A_i}$ denoting the parent of node **i** in tree 1

The third line contains $\mathbf{N_2 - 1}$ space-separated integers $\mathbf{B_2}$, $\mathbf{B_3}$, ..., $\mathbf{B_{N2}}$ with each $\mathbf{B_i}$ denoting the parent of node **i** in tree 2

Output

For each test case, output a single line containing the number of distinct trees obtained by performing the above operation exactly once.

Constraints

- $\mathbf{1 \leq T \leq 10^5}$
- $\mathbf{2 \leq N_1, N_2 \leq 10^5}$
- The sum of $\mathbf{N_1}$ across all test cases does not exceed $\mathbf{10^6}$
- The sum of $\mathbf{N_2}$ across all test cases does not exceed $\mathbf{10^6}$
- $\mathbf{1 \leq A_i, B_i < i}$

Example

```
Input:
2
3 3
1 1
1 1
3 4
1 1
1 1 2

Output:
2
3
```

Explanation

Example case 1. There are two distinct trees that can be formed: the first type is making the entire tree 2 a child of one of the 2 leaves of tree 2 and the other being making one of the leaves of tree 2 a child of one of the leaves of tree 1. Note that which leaf doesn't matter as they can be rearranged to obtain the other way (i.e. if we add it to left leaf, then it is same as adding to the right leaf as we can rearrange the leafs to make it same)

Example case 2. The 3 distinct trees are formed by: make the entire tree 2, subtree rooted at node 2 or any of the leaf of tree 2 the child of a leaf of tree 1

Problem code: **COMPEXP**
Problem name: **Compression Algorithm**

Mr. X has come up with a new string compression algorithm. Consider a string of length **N** which contains up to **K** distinct characters. The compression algorithm works as follows: Replace each maximal contiguous substring containing only one distinct character (repeated an arbitrary number of times) and replace it by 2 values: the character and the length of the substring.

For example, the string "aabbaaa" will be compressed to "a, 2, b, 2, a, 3". Thus the length of the compressed string is 6.

Since Mr. X is living in advanced times, the length of any integer is considered to be 1. For example, if a string is compressed to "a, 111, b, 13", then its length after compression is considered to be 4.

To test his algorithm, he needs to know the expected length of the compressed string for given **N** and **K** if the input string is randomly uniformly chosen from all possibilities. He wants to run this experiment multiple times for different **N, K** and needs your help.

Input

The first line of the input contains an integer **T** denoting the number of queries. The description of **T** test cases follows.

The first and only line of each test case contains two integers **N** and **K** denoting the number of letters in the input string and the maximum number of distinct characters that can be present in the string.

Output

For each test case, output a single line containing the expected length of the compressed string. Your answer will be considered correct if the absolute error is less than $\mathbf{10^{-6}}$

Constraints

- **$1 \le T \le 10^5$**
- **$1 \le N, K \le 10^9$**

Example

```
Input:
2
3 1
3 2
Output:
2.0
4.0
```

Explanation

Example case 1:

There is only one string: aaa with compressed string = a, 3. Therefore length = 2

Example case 2

Input strings:
"aaa": "a, 3". Length = 2
"aab": "a, 2, b, 1". Length = 4
"aba": "a, 1, b, 1, a, 1". Length = 6
"abb": "a, 1, b, 2". Length = 4
"baa": "b, 1, a, 2". Length = 4
"bab": "b, 1, a, 1, b, 1". Length = 6
"bba": "b, 2, a, 1". Length = 4
"bbb": "b, 3". Length = 2
Expected value = **(2+4+6+4+4+6+4+2)/8 = 32/8 = 4**