2020-2021 ICPC, NERC, Southern and Volga Russian Regional Contest (Online Mirror, ICPC Rules)

A. LaIS

time limit per test: 3 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's call a sequence $b_1, b_2, b_3 \dots, b_{k-1}, b_k$ *almost increasing* if

$$\min(b_1, b_2) \le \min(b_2, b_3) \le \dots \le \min(b_{k-1}, b_k).$$

In particular, any sequence with no more than two elements is almost increasing.

You are given a sequence of integers $a_1, a_2, \dots, a_n$. Calculate the length of its longest almost increasing subsequence.

You'll be given t test cases. Solve each test case independently.

Reminder: a *subsequence* is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of independent test cases.

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

The second line of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$) — the sequence itself.

It's guaranteed that the total sum of n over all test cases doesn't exceed $5 \cdot 10^5$.

Output

For each test case, print one integer — the length of the longest almost increasing subsequence.

Example

input

```
3
8
1 2 7 3 2 1 2 3
2
2 1
7
4 1 5 2 6 3 7
```

output

```
6
2
7
```

Note

In the first test case, one of the optimal answers is subsequence $1, 2, 7, 2, 2, 3$.

In the second and third test cases, the whole sequence a is already almost increasing.

B. Bakery

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Monocarp would like to open a bakery in his local area. But, at first, he should figure out whether he can compete with other shops.

Monocarp plans that the bakery will work for n days. On the i-th day, a_i loaves of bread will be baked in the morning before the opening. At the end of the n-th day, Monocarp will sell all the remaining bread that wasn't sold earlier with a huge discount.

Because of how bread is stored, the bakery seller sells the bread in the following order: firstly, he sells the loaves that were baked that morning; secondly, he sells the loaves that were baked the day before and weren't sold yet; then the loaves that were baked two days before and weren't sold yet, and so on. That's why some customers may buy a rather stale bread and will definitely spread negative rumors.

Let's define loaf *spoilage* as the difference between the day it was baked and the day it was sold. Then the *unattractiveness* of the bakery will be equal to the maximum spoilage among all loaves of bread baked at the bakery.

Suppose Monocarp's local area has *consumer demand* equal to k, it means that each day k customers will come to the bakery and each of them will ask for one loaf of bread (the loaves are sold according to the aforementioned order). If there is no bread left, then the person just doesn't buy anything. During the last day sale, all the remaining loaves will be sold (and they will still count in the calculation of the unattractiveness).

Monocarp analyzed his competitors' data and came up with m possible consumer demand values $k_1, k_2, \dots, k_m$, and now he'd like to calculate the unattractiveness of the bakery for each value of demand. Can you help him?

Input

The first line contains two integers n and m ($1 \le n \le 2 \cdot 10^5$; $1 \le m \le 2 \cdot 10^5$) — the number of days the bakery is open and the number of possible values of consumer demand.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the number of bread loaves that will be baked each day.

The third line contains m integers $k_1, k_2, \dots, k_m$ ($1 \le k_1 < k_2 < \dots < k_m \le 10^9$) — the possible consumer demand values in the ascending order.

Output

Print m integers: for each consumer demand, print the unattractiveness of the bakery.

Examples

input

```
5 4
5 2 1 3 7
1 3 4 10
```

output

```
4 2 1 0
```

input

```
8 9
3 1 4 1 5 9 2 6
1 2 3 4 5 6 7 8 9
```

output

```
7 5 3 3 2 1 1 1 0
```

Note

In the first example, let's describe what happens for couple consumer demands:

If consumer demand is equal to 1:

- at day 1: 5 loaves are baked and only 1 is sold with spoilage equal to $1 - 1 = 0$;
- at day 2: 4 loaves are left and 2 more are baked. Only 1 loaf was sold and it was the loaf baked today with spoilage $2 - 2 = 0$;
- at day 3: 4 loaves from the first day and 1 loaf from the second day left. One more loaf was baked and was sold this day with spoilage $3 - 3 = 0$;
- at day 4: 4 loaves from the first day and 1 loaf from the second day left. 3 more loaves were baked and one of them was sold this day with spoilage $4 - 4 = 0$;
- at day 5: 4 loaves from the first day, 1 loaf from the second day and 2 loaves from the fourth day left. 7 more loaves were baked and, since it's the last day, all 14 loaves were sold. 4 loaves from the first day have the maximum spoilage equal to $5 - 1 = 4$.

In total, the unattractiveness of the bakery will be equal to 4.
If consumer demand is equal to 10 then all baked bread will be sold in the day it was baked and will have spoilage equal to 0.

C. Berpizza

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Monocarp and Polycarp are working as waiters in Berpizza, a pizzeria located near the center of Bertown. Since they are waiters, their job is to serve the customers, but they choose whom they serve first differently.

At the start of the working day, there are no customers at the Berpizza. They come there one by one. When a customer comes into the pizzeria, she sits and waits for Monocarp or Polycarp to serve her. Monocarp has been working in Berpizza for just two weeks, so whenever he serves a customer, he simply chooses the one who came to Berpizza first, and serves that customer.

On the other hand, Polycarp is an experienced waiter at Berpizza, and he knows which customers are going to spend a lot of money at the pizzeria (and which aren't) as soon as he sees them. For each customer, Polycarp estimates the amount of money this customer can spend, and when he serves a customer, he chooses the one that is expected to leave the most money at Berpizza (in case there are several such customers, he chooses the one who came first among them).

Obviously, no customer can be served twice, so Monocarp and Polycarp choose which customer to serve only among those who haven't been served yet.

When the number of customers gets really high, it becomes difficult for both Monocarp and Polycarp to choose the customer they are going to serve. Your task is to write a program that makes these choices for them. Formally, your program should be able to process three types of queries:

- $1\ m$ — a customer comes to Berpizza, and Polycarp estimates the amount of money that they will spend as m;
- 2 — Monocarp serves a customer which came to the pizzeria first;
- 3 — Polycarp serves a customer which is expected to spend the largest amount of money at the pizzeria (if there are several such customers, the one that came to the pizzeria first is chosen).

For each query of types 2 and 3, report the number of the customer who was served (the customers are numbered in the order they come to the pizzeria, starting from 1).

Input

The first line contains one integer q ($2 \le q \le 5 \cdot 10^5$) — the number of queries.

Then q lines follow, each describing a query in one of the following formats:

- $1\ m$ ($1 \le m \le 5 \cdot 10^5$) — a customer comes to Berpizza, and Polycarp estimates the amount of money that they will spend as m;
- 2 — Monocarp serves a customer which came to the pizzeria first;
- 3 — Polycarp serves a customer which is expected to spend the largest amount of money at the pizzeria (if there are multiple such customers, the one that came to the pizzeria first is chosen).

Queries of type 2 and 3 are asked only when there exists at least one customer that hasn't been served yet. There is at least one query of type 2 or 3 in the input.

Output

For each query of type 2 or 3, print one integer — the number of the customer that has been served in that event. The customers are numbered in the order in which they come to the pizzeria, starting from 1.

Examples

input

```
8
1 8
1 10
1 6
3
2
1 9
2
3
```

output

```
2 1 3 4
```

input

```
6
1 8
1 10
1 8
3
3
3
```

output

```
2 1 3
```

input

```
8
1 103913
3
1 103913
1 103913
3
1 103913
1 103913
2
```

output

```
1 2 3
```

D. Firecrackers

time limit per test: 4 seconds

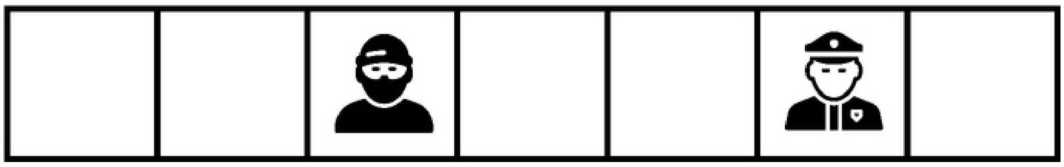
memory limit per test: 512 megabytes

input: standard input

output: standard output

Consider a long corridor which can be divided into n square cells of size 1×1. These cells are numbered from 1 to n from left to right.

There are two people in this corridor, a hooligan and a security guard. Initially, the hooligan is in the a-th cell, the guard is in the b-th cell ($a \neq b$).


One of the possible situations. The corridor consists of 7 cells, the hooligan is in the 3-rd cell, the guard is in the 6-th ($n = 7$, $a = 3$, $b = 6$).

There are m firecrackers in the hooligan's pocket, the i-th firecracker explodes in s_i seconds after being lit.

The following events happen each second (sequentially, **exactly in the following order**):

1. firstly, the hooligan either moves into an adjacent cell (from the cell i, he can move to the cell $(i + 1)$ or to the cell $(i - 1)$, and he cannot leave the corridor) or stays in the cell he is currently. If the hooligan doesn't move, he can light **one** of his firecrackers and drop it. The hooligan can't move into the cell where the guard is;
2. secondly, some firecrackers that were already dropped may explode. Formally, if the firecracker j is dropped on the T-th second, then it will explode on the $(T + s_j)$-th second (for example, if a firecracker with $s_j = 2$ is dropped on the 4-th second, it explodes on the 6-th second);
3. finally, the guard moves one cell closer to the hooligan. If the guard moves to the cell where the hooligan is, the hooligan is caught.

Obviously, the hooligan will be caught sooner or later, since the corridor is finite. His goal is to see the maximum number of firecrackers explode before he is caught; that is, he will act in order to maximize the number of firecrackers that explodes before he is caught.

Your task is to calculate the number of such firecrackers, if the hooligan acts optimally.

Input

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

Each test case consists of two lines. The first line contains four integers n, m, a and b ($2 \le n \le 10^9$; $1 \le m \le 2 \cdot 10^5$; $1 \le a, b \le n$; $a \neq b$) — the size of the corridor, the number of firecrackers, the initial location of the hooligan and the initial location of the guard, respectively.

The second line contains m integers s_1, s_2, ..., s_m ($1 \le s_i \le 10^9$), where s_i is the time it takes the i-th firecracker to explode after it is lit.

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

Output

For each test case, print one integer — the maximum number of firecrackers that the hooligan can explode before he is caught.

Example

input

```
3
7 2 3 6
1 4
7 2 3 6
5 1
7 2 3 6
4 4
```

output

```
2
1
1
```

Note

In the first test case, the hooligan should act, for example, as follows:

- second 1: drop the second firecracker, so it will explode on the 5-th second. The guard moves to the cell 5;
- second 2: move to the cell 2. The guard moves to the cell 4;
- second 3: drop the first firecracker, so it will explode on the 4-th second. The guard moves to the cell 3;
- second 4: move to the cell 1. The first firecracker explodes. The guard moves to the cell 2;
- second 5: stay in the cell 1. The second firecracker explodes. The guard moves to the cell 1 and catches the hooligan.

E. Four Segments

time limit per test: 2 seconds

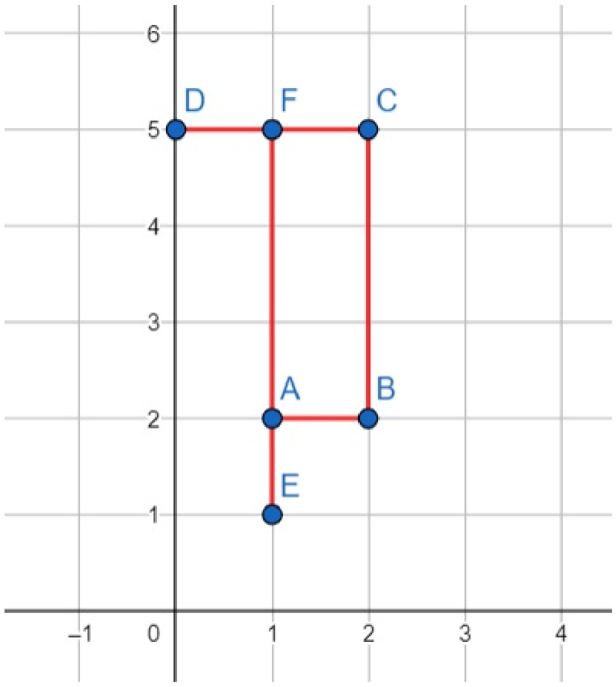
memory limit per test: 512 megabytes

input: standard input
output: standard output

Monocarp wants to draw four line segments on a sheet of paper. He wants the i-th segment to have its length equal to a_i ($1 \le i \le 4$). These segments can intersect with each other, and each segment should be either horizontal or vertical.

Monocarp wants to draw the segments in such a way that they enclose a rectangular space, and the area of that rectangular space should be maximum possible.

For example, if Monocarp wants to draw four segments with lengths 1, 2, 3 and 4, he can do it the following way:



Here, Monocarp has drawn segments AB (with length 1), CD (with length 2), BC (with length 3) and EF (with length 4). He got a rectangle $ABCF$ with area equal to 3 that is enclosed by the segments.

Calculate the maximum area of a rectangle Monocarp can enclose with four segments.

Input

The first line contains one integer t ($1 \le t \le 3 \cdot 10^4$) — the number of test cases.

Each test case consists of a single line containing four integers a_1, a_2, a_3, a_4 ($1 \le a_i \le 10^4$) — the lengths of the segments Monocarp wants to draw.

Output

For each test case, print one integer — the maximum area of a rectangle Monocarp can enclose with four segments (it can be shown that the answer is always an integer).

Example

input

```
4
1 2 3 4
5 5 5 5
3 1 4 1
100 20 20 100
```

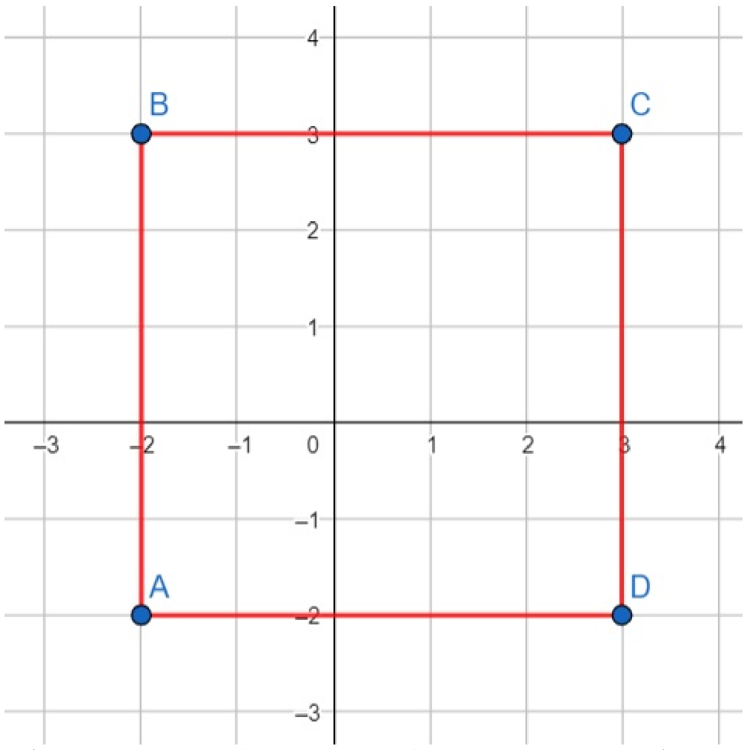
output

```
3
25
3
2000
```

Note

The first test case of the example is described in the statement.

For the second test case, Monocarp can draw the segments AB, BC, CD and DA as follows:



Here, Monocarp has drawn segments AB (with length 5), BC (with length 5), CD (with length 5) and DA (with length 5). He got a rectangle $ABCD$ with area equal to 25 that is enclosed by the segments.

F. Full Turn

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are n persons located on a plane. The i-th person is located at the point (x_i, y_i) and initially looks at the point (u_i, v_i).

At the same moment of time, all persons will start to rotate clockwise synchronously with the same angular speed. They will rotate until they do one full 360-degree turn.

It is said that persons A and B made eye contact if person A looks in person B's direction at the same moment when person B looks in person A's direction. If there is a person C located between persons A and B, that will not obstruct A and B from making eye contact. A person can make eye contact with more than one person at the same time.

Calculate the number of pairs of persons that will make eye contact at least once during the rotation (including the initial moment).

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 10^5$) — the number of persons. The following n lines describe persons, each line containing four space-separated integers x_i, y_i, u_i, v_i ($|x_i|, |y_i|, |u_i|, |v_i| \le 10^9$; $x_i \ne u_i$ or $y_i \ne v_i$), where (x_i, y_i) are the coordinates of the point where the i-th person is located and (u_i, v_i) are the coordinates of the point that the i-th person looks at initially. Each person's location is unique in each test case.

The sum of n over all test cases does not exceed 10^5.

Output

For each test case, print one integer — the number of pairs of persons who will make eye contact at least once during the rotation, including the initial moment.

Example

input

```
3
2
0 0 0 1
1 0 2 0
3
0 0 1 1
1 1 0 0
1 0 2 0
6
0 0 0 1
1 0 1 2
2 0 2 3
3 0 3 -5
4 0 4 -5
5 0 5 -5
```

output

```
0
1
9
```

G. Hobbits

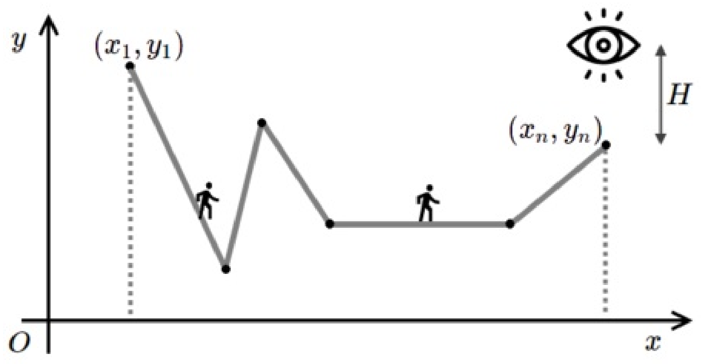
time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

The hobbits Frodo and Sam are carrying the One Ring to Mordor. In order not to be spotted by orcs, they decided to go through the mountains.

The mountain relief can be represented as a polyline with n points (x_i, y_i), numbered from 1 to n ($x_i < x_{i+1}$ for $1 \le i \le n-1$). Hobbits start their journey at the point (x_1, y_1) and should reach the point (x_n, y_n) to complete their mission.

The problem is that there is a tower with the Eye of Sauron, which watches them. The tower is located at the point (x_n, y_n) and has the height H, so the Eye is located at the point $(x_n, y_n + H)$. In order to complete the mission successfully, the hobbits have to wear cloaks all the time when the Sauron Eye can see them, i. e. when there is a direct line from the Eye to the hobbits which is not intersected by the relief.

The hobbits are low, so their height can be considered negligibly small, but still positive, so when a direct line from the Sauron Eye to the hobbits only touches the relief, the Eye can see them.



The Sauron Eye can't see hobbits when they are in the left position, but can see them when they are in the right position.

The hobbits do not like to wear cloaks, so they wear them only when they can be spotted by the Eye. Your task is to calculate the total distance the hobbits have to walk while wearing cloaks.

Input

The first line of the input contains two integers n and H ($2 \le n \le 2 \cdot 10^5$; $1 \le H \le 10^4$) — the number of vertices in polyline and the tower height.

The next n lines contain two integers x_i, y_i each ($0 \le x_i \le 4 \cdot 10^5$; $0 \le y_i \le 10^4$) — the coordinates of the polyline vertices. It is guaranteed that $x_i < x_{i+1}$ for $1 \le i \le n-1$.

Output

Print one real number — the total distance the hobbits have to walk while wearing cloaks. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-6} — formally, if your answer is a, and the jury's answer is b, your answer will be accepted if $\frac{|a-b|}{\max(1,b)} \le 10^{-6}$.

Examples

input

```
6 10
10 40
20 10
25 30
30 15
50 15
65 30
```

output

```
70.4034587602
```

input

```
9 5
0 0
5 10
15 10
20 0
25 11
```

```
30 0
35 10
50 10
60 5
```

output

```
27.2787986124
```

input

```
2 10000
0 10000
400000 0
```

output

```
400124.9804748512
```

H. K and Medians

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's denote the *median* of a sequence s with odd length as the value in the middle of s if we sort s in non-decreasing order. For example, let $s = [1, 2, 5, 7, 2, 3, 12]$. After sorting, we get sequence $[1, 2, 2, \underline{3}, 5, 7, 12]$, and the median is equal to 3.

You have a sequence of n integers $[1, 2, \ldots, n]$ and an **odd** integer k.

In one step, you choose any k elements from the sequence and erase all chosen elements **except** their median. These elements do not have to go continuously (gaps are allowed between them).

For example, if you have a sequence $[1, 2, 3, 4, 5, 6, 7]$ (i.e. $n = 7$) and $k = 3$, then the following options for the first step are possible:

- choose $[1, \underline{2}, 3]$; 2 is their median, so it is not erased, and the resulting sequence is $[2, 4, 5, 6, 7]$;
- choose $[2, \underline{4}, 6]$; 4 is their median, so it is not erased, and the resulting sequence is $[1, 3, 4, 5, 7]$;
- choose $[1, \underline{6}, 7]$; 6 is their median, so it is not erased, and the resulting sequence is $[2, 3, 4, 5, 6]$;
- and several others.

You can do zero or more steps. Can you get a sequence b_1, b_2, ..., b_m after several steps?

You'll be given t test cases. Solve each test case independently.

Input

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

The first line of each test case contains three integers n, k, and m ($3 \le n \le 2 \cdot 10^5$; $3 \le k \le n$; k is **odd**; $1 \le m < n$) — the length of the sequence you have, the number of elements you choose in each step and the length of the sequence you'd like to get.

The second line of each test case contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_1 < b_2 < \cdots < b_m \le n$) — the sequence you'd like to get, given in the ascending order.

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

Output

For each test case, print YES if you can obtain the sequence b or `NO` otherwise. You may print each letter in any case (for example, YES, Yes, yes, yEs will all be recognized as positive answer).

Example

input

```
4
3 3 1
1
7 3 3
1 5 7
10 5 3
4 5 6
13 7 7
1 3 5 7 9 11 12
```

output

```
NO
YES
NO
YES
```

Note

In the first test case, you have sequence $[1, 2, 3]$. Since $k = 3$ you have only one way to choose k elements — it's to choose all elements $[1, \underline{2}, 3]$ with median 2. That's why after erasing all chosen elements except its median you'll get sequence $[2]$. In other words, there is no way to get sequence $b = [1]$ as the result.

In the second test case, you have sequence $[1, 2, 3, 4, 5, 6, 7]$ and one of the optimal strategies is following:

1. choose $k = 3$ elements $[2, \underline{3}, 4]$ and erase them except its median; you'll get sequence $[1, 3, 5, 6, 7]$;
2. choose 3 elements $[3, \underline{5}, 6]$ and erase them except its median; you'll get desired sequence $[1, 5, 7]$;

In the fourth test case, you have sequence $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]$. You can choose $k = 7$ elements $[2, 4, 6, \underline{7}, 8, 10, 13]$ and erase them except its median to get sequence b.

I. Plane Tiling

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given five integers n, dx_1, dy_1, dx_2 and dy_2. You have to select n distinct pairs of integers (x_i, y_i) in such a way that, for every possible pair of integers (x, y), there exists exactly one triple of integers (a, b, i) meeting the following constraints:

$$\begin{cases} x = x_i + a \cdot dx_1 + b \cdot dx_2, \\ y = y_i + a \cdot dy_1 + b \cdot dy_2. \end{cases}$$

Input

The first line contains a single integer n ($1 \le n \le 10^5$).

The second line contains two integers dx_1 and dy_1 ($-10^6 \le dx_1, dy_1 \le 10^6$).

The third line contains two integers dx_2 and dy_2 ($-10^6 \le dx_2, dy_2 \le 10^6$).

Output

If it is impossible to correctly select n pairs of integers, print NO.

Otherwise, print YES in the first line, and then n lines, the i-th of which contains two integers x_i and y_i ($-10^9 \le x_i, y_i \le 10^9$).

If there are multiple solutions, print any of them.

Examples

input
```
4
2 0
0 2
```
output
```
YES
0 0
0 1
1 0
1 1
```

input
```
5
2 6
1 5
```
output
```
NO
```

input
```
2
3 4
1 2
```
output
```
YES
0 0
0 1
```

J. Road Reform

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There are n cities and m bidirectional roads in Berland. The i-th road connects the cities x_i and y_i, and has the speed limit s_i. The road network allows everyone to get from any city to any other city.

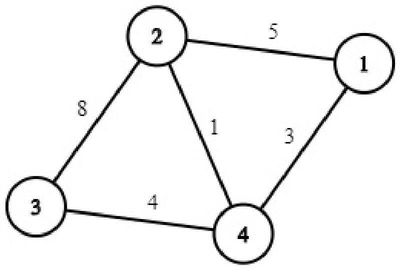
The Berland Transport Ministry is planning a road reform.

First of all, maintaining all m roads is too costly, so $m-(n-1)$ roads will be demolished in such a way that the remaining $(n-1)$ roads still allow to get to any city from any other city. Formally, the remaining roads should represent an undirected tree.

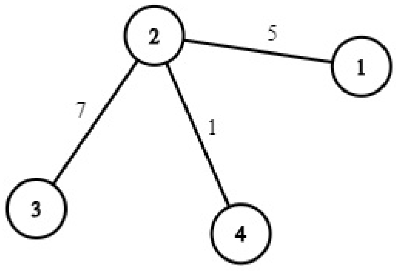
Secondly, the speed limits on the remaining roads might be changed. The changes will be done sequentially, each change is either increasing the speed limit on some road by 1, or decreasing it by 1. Since changing the speed limit requires a lot of work, the Ministry wants to minimize the number of changes.

The goal of the Ministry is to have a road network of $(n-1)$ roads with the maximum speed limit over all roads equal to exactly k. They assigned you the task of calculating the minimum number of speed limit changes they have to perform so the road network meets their requirements.

For example, suppose the initial map of Berland looks like that, and $k=7$:



Then one of the optimal courses of action is to demolish the roads 1–4 and 3–4, and then decrease the speed limit on the road 2–3 by 1, so the resulting road network looks like that:



Input

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

The first line of each test case contains three integers n, m and k ($2 \le n \le 2 \cdot 10^5$; $n-1 \le m \le \min(2 \cdot 10^5, \frac{n(n-1)}{2})$; $1 \le k \le 10^9$) — the number of cities, the number of roads and the required maximum speed limit, respectively.

Then m lines follow. The i-th line contains three integers x_i, y_i and s_i ($1 \le x_i, y_i \le n$; $x_i \ne y_i$; $1 \le s_i \le 10^9$) — the cities connected by the i-th road and the speed limit on it, respectively. All roads are bidirectional.

The road network in each test case is connected (that is, it is possible to reach any city from any other city by traveling along the road), and each pair of cities is connected by at most one road.

The sum of n over all test cases does not exceed $2 \cdot 10^5$. Similarly, the sum of m over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print one integer — the minimum number of changes the Ministry has to perform so that the maximum speed limit among the remaining $(n-1)$ roads is exactly k.

Example

input

```
4
4 5 7
4 1 3
1 2 5
2 3 8
2 4 1
3 4 4
4 6 5
1 2 1
1 3 1
1 4 2
2 4 1
4 3 1
3 2 1
3 2 10
```

```
1 2 8
1 3 10
5 5 15
1 2 17
3 1 15
2 3 10
1 4 14
2 5 8
```

output

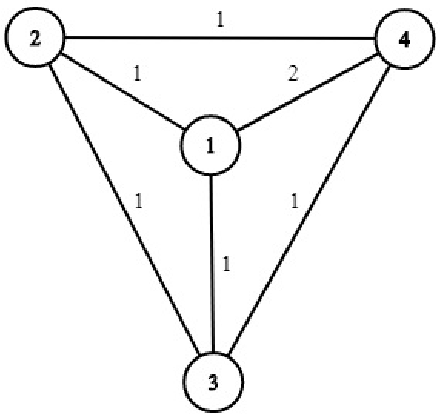
```
1
3
0
0
```

Note

The explanation for the example test:

The first test case is described in the problem statement.

In the second test case, the road network initially looks like that:



The Ministry can demolish the roads 1–2, 3–2 and 3–4, and then increase the speed limit on the road 1–4 three times.

In the third test case, the road network already meets all the requirements.

In the fourth test case, it is enough to demolish the road 1–2 so the resulting road network meets the requirements.

K. The Robot

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

There is a robot on a checkered field that is endless in all directions. Initially, the robot is located in the cell with coordinates $(0, 0)$. He will execute commands which are described by a string of capital Latin letters 'L', 'R', 'D', 'U'. When a command is executed, the robot simply moves in the corresponding direction:

- 'L': one cell to the left (the x-coordinate of the current cell decreases by 1);
- 'R': one cell to the right (the x-coordinate of the current cell is increased by 1);
- 'D': one cell down (the y-coordinate of the current cell decreases by 1);
- 'U': one cell up (the y-coordinate of the current cell is increased by 1).

Your task is to put an obstacle in one cell of the field so that after executing the commands, the robot will return to the original cell of its path $(0, 0)$. Of course, an obstacle cannot be placed in the starting cell $(0, 0)$. It is guaranteed that if the obstacle is not placed, then the robot will not return to the starting cell.

An obstacle affects the movement of the robot in the following way: if it tries to go in a certain direction, and there is an obstacle, then it simply remains in place (the obstacle also remains, that is, it does not disappear).

Find any such cell of the field (other than $(0, 0)$) that if you put an obstacle there, the robot will return to the cell $(0, 0)$ after the execution of all commands. If there is no solution, then report it.

Input

The first line contains one integer t ($1 \le t \le 500$) — the number of test cases.

Each test case consists of a single line containing s — the sequence of commands, which are uppercase Latin letters 'L', 'R', 'D', 'U' only. The length of s is between 1 and 5000, inclusive. Additional constraint on s: executing this sequence of commands leads the robot to some cell other than $(0, 0)$, if there are no obstacles.

The sum of lengths of all s in a test doesn't exceed 5000.

Output

For each test case print a single line:

- if there is a solution, print two integers x and y ($-10^9 \le x, y \le 10^9$) such that an obstacle in (x, y) will force the robot to return back to the cell $(0, 0)$;
- otherwise, print two zeroes (i. e. `0 0`).

If there are multiple answers, you can print any of them.

Example

input

```
4
L
RUUDL
LLUU
DDDUUUUU
```

output

```
-1 0
1 2
0 0
0 1
```

L. Prime Divisors Selection

time limit per test: 2 seconds

memory limit per test: 512 megabytes

input: standard input

output: standard output

Suppose you have a sequence of k integers $A = [a_1, a_2, \dots, a_k]$ where each $a_i \geq 2$. A sequence of **prime** integers $P = [p_1, p_2, \dots, p_k]$ is called *suitable* for the sequence A if a_1 is divisible by p_1, a_2 is divisible by p_2 and so on.

A sequence of **prime** integers P is called *friendly* if there are no unique integers in this sequence.

A sequence A is called *ideal*, if each sequence P that is *suitable* for A is *friendly* as well (i. e. there is no sequence P that is *suitable* for A, but not *friendly*). For example, the sequence $[2, 4, 16]$ is *ideal*, while the sequence $[2, 4, 6]$ is not *ideal* (there exists a sequence $P = [2, 2, 3]$ which is *suitable* for A, but not *friendly*).

You are given n different integers x_1, x_2, ..., x_n. You have to choose **exactly** k of them in such a way that they form an *ideal* sequence, or report that it is impossible. Note that no integer can be chosen more than once.

Input

The first line contains two integers n and k ($1 \le k \le n \le 1000$).

The second line contains n pairwise distinct integers x_1, x_2, ..., x_n ($2 \le x_i \le 10^{18}$).

Output

If it is impossible to choose exactly k integers from x_1, x_2, ..., x_n in such a way that the chosen integers form an *ideal* sequence, print 0.

Otherwise, print k pairwise distinct integers — the elements of the chosen *ideal* sequence. If there are multiple answers, print any of them.

Examples

input

```
3 3
2 4 6
```

output

```
0
```

input

```
3 3
2 4 16
```

output

```
2 4 16
```

input

```
4 3
2 4 6 16
```

output

```
2 4 16
```

M. Similar Sets

time limit per test: 1 second
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given n sets of integers. The i-th set contains k_i integers.

Two sets are called *similar* if they share at least two common elements, i. e. there exist two integers x and y such that $x \neq y$, and they both belong to each of the two sets.

Your task is to find two similar sets among the given ones, or report that there is no such pair of sets.

Input

The first line contains a single integer t ($1 \leq t \leq 50000$) — the number of test cases. Then t test cases follow.

The first line of each test case contains a single integer n ($2 \leq n \leq 10^5$) the number of given sets. The following n lines describe the sets. The i-th line starts with an integer k_i ($2 \leq k_i \leq 10^5$) — the number of integers in the i-th set. Then k_i integers $a_{i,1}$, $a_{i,2}$, ..., a_{i,k_i} ($1 \leq a_{i,j} \leq 10^9$) follow — the elements of the i-th set. It is guaranteed that all elements in each set are different.

The total number of elements in all sets in all test cases is not greater than $2 \cdot 10^5$.

Output

For each test case, print the answer on a single line.

If there is no pair of *similar* sets, print -1.

Otherwise, print two different integers — the indices of the *similar* sets. The sets are numbered from 1 to n in the order they are given in the input. If there are multiple answers, print any of them.

Example

input

```
3
4
2 1 10
3 1 3 5
5 5 4 3 2 1
3 10 20 30
3
4 1 2 3 4
4 2 3 4 5
4 3 4 5 6
2
3 1 3 5
3 4 3 2
```

output

```
2 3
1 2
-1
```

N. Waste Sorting

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

The progress is not standing still in Berland. Recently all garbage containers in Bertown, the capital of Berland, were replaced by differentiated recycling bins, each accepting some category of waste. While this will definitely improve the ecological situation, for some citizens it's difficult to get used to the habit of sorting waste.

Monocarp is one of those citizens who tries to get used to waste sorting. Today he has to take out the trash from his house. There are three containers near the Monocarp's house, the first one accepts paper waste, the second one accepts plastic waste, and the third one — all other types of waste. It is possible to fit c_1 items into the first container, c_2 items into the second container and c_3 items into the third container.

Monocarp has a lot of items to throw into containers. Some are made of paper, so Monocarp has to put them into the first container (he has a_1 such items), some are made of plastic, so he has to put them into the second container (he has a_2 such items), and some are neither paper nor plastic — so Monocarp has to put them into the third container (he has a_3 such items).

Unfortunately, there are also two categories of items that Monocarp is unsure of: he has a_4 items which are partially made of paper, so he will put each of these items either into the first container or into the third container. Similarly, he has a_5 items partially made of plastic, so he has to put each of them either into the second container or into the third container. Obviously, this choice is made separately for each item — for example, Monocarp can throw several partially-plastic items into the second container, and all other partially-plastic items — into the third one.

Now Monocarp wonders: is it possible to put each item into some container so that the first container will hold no more than c_1

items, the second one — no more than c_2 items, and the third one — no more than c_3 items?

Input

The first line contains one integer t ($1 \le t \le 3 \cdot 10^4$) — the number of test cases.

Each test case consists of two lines. The first line of each test case contains three integers c_1, c_2, c_3 ($0 \le c_1, c_2, c_3 \le 10^8$) — the capacities of the containers.

The second line of each test case contains five integers a_1, a_2, a_3, a_4, a_5 ($0 \le a_i \le 10^8$), where a_i is the number of items of the i-th category Monocarp has to throw out ($i = 1$ is paper waste, $i = 2$ is plastic waste, $i = 3$ is general waste, $i = 4$ is partially-paper waste, $i = 5$ is partially-plastic waste).

Output

For each test case, print either YES if it is possible to fit all items into containers, or NO otherwise. You may print each letter in any case (for example, YES, Yes, yes, yEs will all be recognized as positive answer).

Example

input

```
7
1 2 3
1 2 3 0 0
2 2 3
1 2 3 1 0
2 2 3
1 2 3 0 1
1 2 5
1 2 3 1 1
0 0 0
0 0 0 0 0
0 0 4
1 0 0 0 0
13 37 42
0 0 0 40 47
```

output

```
YES
YES
NO
YES
YES
NO
YES
```

Note

Explanations for the example test cases:

1. Monocarp can put 1 item of paper waste into the first container, 2 items of plastic waste into the second container, and 3 items of general waste into the third container;
2. Monocarp can put 1 item of paper waste and 1 item of partially-paper waste into the first container, 2 items of plastic waste into the second container, and 3 items of general waste into the third container;
3. there is no answer since either the second container should contain 3 items, or the third container should contain 4 items;
4. Monocarp can put 1 item of paper waste into the first container, 2 items of plastic waste into the second container, and 3 items of general waste, 1 item of partially-paper waste and 1 item of partially-plastic waste into the third container;
5. there is no waste at all, so all containers can be left empty;
6. there's no answer since it's impossible to put a paper item into the third container;
7. Monocarp can put 10 items of partially-paper waste into the first container, 37 items of partially-plastic waste into the second container, and 30 items of partially-paper waste and 10 items of partially-plastic waste into the third container.