Problem A. Armor and Weapons

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Monocarp plays a computer game. There are n different sets of armor and m different weapons in this game. If a character equips the i-th set of armor and wields the j-th weapon, their power is usually equal to $i+j$; but some combinations of armor and weapons synergize well. Formally, there is a list of q ordered pairs, and if the pair (i,j) belongs to this list, the power of the character equipped with the i-th set of armor and wielding the j-th weapon is not $i+j$, but $i+j+1$.

Initially, Monocarp's character has got only the 1-st armor set and the 1-st weapon. Monocarp can obtain a new weapon or a new set of armor in one hour. If he wants to obtain the k-th armor set or the k-th weapon, he must possess a combination of an armor set and a weapon that gets his power to k **or greater**. Of course, after Monocarp obtains a weapon or an armor set, he can use it to obtain new armor sets or weapons, but he can go with any of the older armor sets and/or weapons as well.

Monocarp wants to obtain the n-th armor set **and** the m-th weapon. What is the minimum number of hours he has to spend on it?

Input

The first line contains two integers n and m ($2 \le n, m \le 2 \cdot 10^5$) — the number of armor sets and the number of weapons, respectively.

The second line contains one integer q ($0 \le q \le \min(2 \cdot 10^5, nm)$) — the number of combinations that synergize well.

Then q lines follow, the i-th line contains two integers a_i and b_i ($1 \le a_i \le n$; $1 \le b_i \le m$) meaning that the a_i-th armor set synergizes well with the b_i-th weapon. All pairs (a_i, b_i) are distinct.

Output

Print one integer — the minimum number of hours Monocarp has to spend to obtain **both** the n-th armor set and the m-th weapon.

Examples

standard input	standard output
3 4 0	3

3 4
2
1 1
1 3
	2

Note

In the first example, Monocarp can obtain the strongest armor set and the strongest weapon as follows:

1. Obtain the 2-nd weapon using the 1-st armor set and the 1-st weapon;
2. Obtain the 3-rd armor set using the 1-st armor set and the 2-nd weapon;
3. Obtain the 4-th weapon using the 3-rd armor set and the 2-nd weapon.

In the second example, Monocarp can obtain the strongest armor set and the strongest weapon as follows:

1. Obtain the 3-rd armor set using the 1-st armor set and the 1-st weapon **(they synergize well, so Monocarp's power is not** 2 **but** 3);
2. Obtain the 4-th weapon using the 3-rd armor set and the 1-st weapon.

Problem B. Special Permutation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

A permutation of length n is an array $p = [p_1, p_2, \dots, p_n]$ which contains every integer from 1 to n (inclusive) exactly once. For example, $p = [4, 2, 6, 5, 3, 1]$ is a permutation of length 6.

You are given three integers n, a and b, where n is an even number. Print any permutation of length n that the minimum among **all its elements of the left half** equals a and the maximum among **all its elements of the right half** equals b. Print `-1` if no such permutation exists.

Input

The first line of the input contains one integer t ($1 \le t \le 1000$), the number of test cases in the test. The following t lines contain test case descriptions.

Each test case description contains three integers n, a, b ($2 \le n \le 1000$; $1 \le a, b \le n$; $a \ne b$), where n is an even number (i.e. $n \bmod 2 = 0$).

Output

For each test case, print a single line containing any suitable permutation. Print `-1` no such permutation exists. If there are multiple answers, print any of them.

Example

`standard input`	`standard output`
`7`	`4 2 6 5 3 1`
`6 2 5`	`-1`
`6 1 3`	`6 4 5 1 3 2`
`6 4 3`	`3 2 4 1`
`4 2 4`	`-1`
`10 5 3`	`1 2`
`2 1 2`	`2 1`
`2 2 1`	

Problem C. Athletes

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

A wrestling club trains n athletes. Each of them specializes either in *algorithmic sambo* (referred below as sport A), or in *berland jiu-jitsu* (referred below as sport B). In addition, each athlete has a certain level of skill.

Thus, the i-th athlete is characterized by two parameters: t_i — a sport (either A or B) and an integer skill level f_i ($-10^6 \le f_i \le 10^6$).

If an athlete from sport A competes in B, then their skill level is reduced by a given number x (the value does not depend on a particular athlete). Similarly, if an athlete from sport B competes in A, then their skill level is reduced by a given number y (again, the value does not depend on a particular athlete).

The club plans to participate in a competition. To participate, it is required to set up k ($2k \le n$) teams of two athletes in each — one of them will compete in sport A, the other will compete in sport B. Of course, no athlete can be included in more than one team.

Your task is to compose k teams in such a way that the total skill of all participating athletes in these teams is maximum possible. Formally, it is necessary to form k teams from the n given athletes, in each team there should be one athlete competing in A and one athlete competing in B. If an athlete competes in a sport they don't specialize in, then their skill decreases by x (if an athlete with $t_i = \text{A}$ competes in B) or by y (if an athlete with $t_i = \text{B}$ competes in A).

Print the maximum possible sum of skill levels of athletes in k teams if the club forms teams in an optimal way. Remember that an athlete's skill level is reduced (by x or y depending on the sport) if they are playing in a sport that is different from their specialization.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input. The following are t test cases.

The first line of each test case contains four integers n, k, x, y ($2 \le n \le 2 \cdot 10^5$; $1 \le k \le \frac{n}{2}$; $1 \le x, y \le 10^6$), where:

- n is the number of athletes in the club,
- k is the required number of teams,
- x is how much the skill level of an athlete of sport A decreases if they compete in sport B,
- y is how much the skill level of an athlete of sport B decreases if they compete in sport A.

Each of the next n lines contains a value t_i (either the character `A` or the character `B`) and an integer f_i ($-10^6 \le f_i \le 10^6$) — the sport of the i-th athlete and their skill level, respectively.

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

Output

For each of the t test cases, print one integer — the maximum total skill of the athletes of k teams, if in each team one athlete will compete in sport A, and the other will compete in sport B. Remember that if an athlete competes in a sport they don't specialize in, then their skill level is reduced.

Example

standard input	standard output
4 4 1 1 2 A 10 B 10 B 20 B 15 4 1 1 100 A 10 B 10 B 20 B 15 4 2 1 1 A 10 B 10 B 20 B 15 4 2 1 23 B 1 B 9 B -3 A 5	33 30 54 -11

Problem D. Max Sum Array

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

You are given an array $c = [c_1, c_2, \dots, c_m]$. An array $a = [a_1, a_2, \dots, a_n]$ is constructed in such a way that it consists of integers $1, 2, \dots, m$, and for each $i \in [1, m]$, there are exactly c_i occurrences of integer i in a. So, the number of elements in a is exactly $\sum\limits_{i=1}^{m} c_i$.

Let's define for such array a the value $f(a)$ as

$$f(a) = \sum_{\substack{1 \le i < j \le n \\ a_i = a_j}} j - i.$$

In other words, $f(a)$ is the total sum of distances between all pairs of equal elements.

Your task is to calculate the maximum possible value of $f(a)$ and the number of arrays yielding the maximum possible value of $f(a)$. Two arrays are considered different, if elements at some position differ.

Input

The first line contains a single integer m ($1 \le m \le 5 \cdot 10^5$) — the size of the array c.

The second line contains m integers $c_1, c_2, \dots, c_m$ ($1 \le c_i \le 10^6$) — the array c.

Output

Print two integers — the maximum possible value of $f(a)$ and the number of arrays a with such value. Since both answers may be too large, print them modulo $10^9 + 7$.

Examples

standard input	standard output
6 1 1 1 1 1 1	0 720
1 1000000	499833345 1
7 123 451 234 512 345 123 451	339854850 882811119

Note

In the first example, all possible arrays a are permutations of $[1, 2, 3, 4, 5, 6]$. Since each array a will have $f(a) = 0$, so maximum value is $f(a) = 0$ and there are $6! = 720$ such arrays.

In the second example, the only possible array consists of 10^6 ones and its $f(a) = \sum\limits_{1 \le i < j \le 10^6} j - i = 166\,666\,666\,666\,500\,000$ and $166\,666\,666\,666\,500\,000 \bmod 10^9 + 7 = 499\,833\,345$.

Problem E. Request Throttling

Input file:	`standard input`
Output file:	`standard output`
Time limit:	6 seconds
Memory limit:	512 megabytes

Modern web systems have several layers of protection from attacks. One of such layers is request throttling, which is aimed to keep incoming request rate at a certain level and, therefore, protect the system from DDoS attacks.

Further on, we will consider a simplified IP-based throttling model. In this problem, we will work with IPv4 addresses only. Let's also assume that there are no reserved or internal addresses and the entire IPv4 address space is used.

The next five paragraphs give the formal definition of *IPv4 address* and *IPv4 subnet*, respectively. They also describe the structure of the IPv4 address space. These are standard definitions and properties, so if you are familiar with them, you can skip those paragraphs.

An IPv4 address is a 32-bit unsigned integer written in the form $a.b.c.d$, where each of the values a, b, c, d is called an octet and is an integer from 0 to 255 written in decimal notation. For example, the IPv4 address 192.168.0.1 can be converted to a 32-bit number using the following expression: $192 \cdot 2^{24} + 168 \cdot 2^{16} + 0 \cdot 2^8 + 1 \cdot 2^0$. The first octet a encodes the most significant (leftmost) 8 bits, the octets b and c — the following blocks of 8 bits (in this order), and the octet d encodes the least significant (rightmost) 8 bits.

An IPv4 subnet of size $/x$ is represented as $a.b.c.d/x$ (where $0 \le x \le 32$). A subnet $a.b.c.d/x$ contains all IPv4 addresses with x leftmost (most significant) bits equal to x leftmost bits of the address $a.b.c.d$. It is required that $32 - x$ rightmost (least significant) bits of subnet $a.b.c.d/x$ are zeroes.

Naturally, it happens that all addresses matching subnet $a.b.c.d/x$ form a continuous range. The range starts with the address $a.b.c.d$ (its rightmost $32 - x$ bits are zeroes). The range ends with the address such that its x leftmost bits equal to x leftmost bits of address $a.b.c.d$, and all its $32 - x$ rightmost bits are ones. A subnet of size $/x$ contains exactly 2^{32-x} addresses.

For example:

- The subnet 192.168.0.0/24 contains a range of 256 addresses. 192.168.0.0 is the first address of the range, and 192.168.0.255 is the last one.
- The subnet 172.168.0.70/31 contains two addresses: 172.168.0.70 and 172.168.0.71.
- The subnet 10.8.179.3/32 contains only one address: 10.8.179.3.

The entire IPv4 address space consists of 2^{32} IP addresses. In the terms of subnets, it means there are 2^{32} different /32 subnets (because each /32 subnet contains exactly one IP address), 2^{31} different /31 subnets (each /31 subnet contains two addresses), and so on, $2^1 = 2$ different /1 subnets, and $2^0 = 1$ subnet of size /0. Furthermore, it can be noted that each IP address is contained in exactly one subnet of each size from /0 to /32 inclusive. Further on, such subnets will be called *parent subnets*. E.g. these are some of the *parent subnets* of the IPv4 address 192.168.21.5: 192.168.21.5/32, 192.168.21.4/30, 192.168.21.0/24, 192.168.16.0/21, 192.168.0.0/16, 192.0.0.0/8, 128.0.0.0/1, 0.0.0.0/0.

The request throttling model is described by 33 integers $t_0, t_1, \ldots, t_{32}$ — the throttling limits for $/0, /1, \ldots, /32$ subnets, respectively. In other words, t_j is the maximum number of requests from the certain subnet of size $/j$ which are allowed to bypass the throttling subsystem and reach other layers of the system. The throttling limit for each of 2^j subnets of size $/j$ is equal to the same number t_j, but passed request count is calculated separately for each individual subnet.

Incoming requests are processed one by one. When a request arrives, its source IP address is analyzed, and if the address is within the throttling limits of all 33 of its parent subnets, then the request is allowed.

It goes through the throttling system, increasing the passed request count for all its parent subnets. Otherwise, the request is blocked and the passed request count for no subnet is increased. We assume that a request is within the throttling limits of a subnet if and only if the passed request count of that subnet is strictly less than the throttling limit for the subnet (i.e. the value of t_j for the corresponding size of the subnet).

Your task is to emulate the work of the throttling subsystem. You are given n incoming requests, where each request is characterized by its source IP address s_i. The requests should be processed in the order they are given. For each request, print "a" (without quotes) if the request is allowed, and "b" (also without quotes) if the request is blocked.

Input

The first line of the input file contains 33 integers $t_0, t_1, \ldots, t_{32}$ — the throttling limits for $/0, /1, \ldots, /32$ subnets, respectively: $1 \le t_j \le 10^9$ for $0 \le j \le 32$.

The second line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of requests. The following n lines describe the source addresses of the requests. It is guaranteed that each source address represents an IPv4 address in the correct format. It is assumed that there are no reserved IPv4 addresses and the entire IPv4 address space can be used.

Output

Output file should contain exactly n lines, the i-th line should contain a single lowercase letter "a" (without quotes) if the i-th request is allowed, or a single lowercase letter "b" (without quotes) if the i-th request is blocked.

Examples

standard input

10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 3 2
5
192.168.0.1
192.168.0.1
192.168.0.1
192.168.0.0
192.168.0.0

standard output
a a b a b

standard input

```
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 10 10 10 10 10 10 10 2 10 10 10 10 10 10 10 10
6
5.5.5.0
5.5.5.1
5.5.5.2
5.5.255.255
5.5.255.0
5.5.255.128
```

standard output

```
a
a
b
a
a
b
```

Note

In both examples, the first line with 33 numbers can't fit on a printed sheet. For this reason, the line is split in two. In the tests on which your submission will be tested, exactly 33 numbers are given in the first line.

In the first example, only limits for /31 and /32 subnets actually influence the throttling, since limits for other subnets are greater than the total number of requests. The first two requests from 192.168.0.1 are allowed, as they are within the limits for 192.168.0.1/32 and 192.168.0.0/31 subnets. The third request from 192.168.0.1 is within the limits for 192.168.0.0/31 subnet (2 out of 3 requests were passed, one more still can be passed), but already exceeds the limits for 192.168.0.1/32 (2 out of 2 requests were already passed). Since the request exceeds the limit for at least one subnet, it is blocked, without affecting the limits for other requests. The next request from 192.168.0.0 is within the limits both for 192.168.0.0/31 and 192.168.0.0/32 subnets, so it is allowed. The last request from 192.168.0.0 is within the limits for 192.168.0.0/32 subnet (1 out of 2 requests are passed), but the limit for 192.168.0.0/31 subnet is already exceeded, so the request is blocked.

The second example demonstrates that two /24 subnets (5.5.5.0/24 and 5.5.255.0/24) are independent, and requests are counted separately for them.

Problem F. X-Magic Pair

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given a pair of integers (a, b) and an integer x.

You can change the pair in two different ways:

- set (assign) $a := |a - b|$;
- set (assign) $b := |a - b|$,

where $|a - b|$ is the absolute difference between a and b.

The pair (a, b) is called x-magic if x is obtainable either as a or as b using only the given operations (i.e. the pair (a, b) is x-magic if $a = x$ or $b = x$ after some number of operations applied). You can apply the operations any number of times (even zero).

Your task is to find out if the pair (a, b) is x-magic or not.

You have to answer t independent test cases.

Input

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

The i-th test case is given as three space-separated integers a, b and x ($1 \le a, b, x \le 10^{18}$).

Output

For the i-th test case, print `YES` if the corresponding pair (a, b) is x-magic and `NO` otherwise.

Example

standard input
```
8
6 9 3
15 38 7
18 8 8
30 30 30
40 50 90
24 28 20
365 216 52
537037812705867558 338887693834423551 3199921013340
```

standard output
```
YES
YES
YES
YES
NO
YES
YES
YES
```

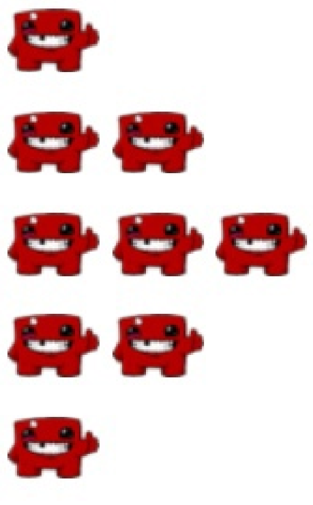
Problem G. Chat Ban

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

You are a usual chat user on the most famous streaming platform. Of course, there are some moments when you just want to chill and spam something.

More precisely, you want to spam the emote triangle of size k. It consists of $2k - 1$ messages. The first message consists of one emote, the second one — of two emotes, …, the k-th one — of k emotes, the $k + 1$-th one — of $k - 1$ emotes, …, and the last one — of one emote.

For example, the emote triangle for $k = 3$ consists of 5 messages:



Of course, most of the channels have auto moderation. Auto moderator of the current chat will ban you right after you spam at least x emotes in succession (you can assume you are the only user in the chat). Now you are interested — how many messages will you write before getting banned? Or maybe you will not get banned at all (i.e. will write all $2k - 1$ messages and complete your emote triangle successfully)? Note that if you get banned as a result of writing a message, this message is counted.

You have to answer t independent test cases.

Input

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

The i-th test case is given as two space-separated integers k and x ($1 \le k \le 10^9; 1 \le x \le 10^{18}$).

Output

For the i-th test case, print the number of messages you will write before getting banned for the corresponding values k and x.

Example

standard input	standard output

7
4 6
4 7
1 2
3 7
2 5
100 1
1000000000 923456789987654321
	3
	4
	1
	4
	3
	1
	1608737403

Note

Let's analyze the test cases of the example.

1. In the first test case, you write three messages containing 1, 2 and 3 emotes respectively, and since $1 + 2 + 3 \geq 6$, you get banned after that.
2. In the second test case, you write four messages containing 1, 2, 3 and 4 emotes respectively, and since $1 + 2 + 3 + 4 \geq 7$, you get banned after that.
3. In the third test case, you write one message containing exactly 1 emote. It doesn't get you banned, since $1 < 2$, but you have already finished posting your emote triangle.
4. In the fourth test case, you write four messages containing 1, 2, 3 and 2 emotes respectively, and since $1 + 2 + 3 + 2 \geq 7$, you get banned after that.
5. In the fifth test case, you write three messages containing 1, 2 and 1 emote respectively. It doesn't get you banned, since $1 + 2 + 1 < 5$, but you have already finished posting your emote triangle.
6. In the sixth test case, since $x = 1$, you get banned as soon as you send your first message.
7. The seventh test case is too large to analyze, so we'll skip it.

Problem H. Messages

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

Monocarp is a tutor of a group of n students. He communicates with them using a conference in a popular messenger.

Today was a busy day for Monocarp — he was asked to forward a lot of posts and announcements to his group, that's why he had to write a very large number of messages in the conference. Monocarp knows the students in the group he is tutoring quite well, so he understands which message should each student read: Monocarp wants the student i to read the message m_i.

Of course, no one's going to read all the messages in the conference. That's why Monocarp decided to pin some of them. Monocarp can pin any number of messages, and if he wants anyone to read some message, he should pin it — otherwise **it will definitely be skipped by everyone**.

Unfortunately, even if a message is pinned, some students may skip it anyway. For each student i, Monocarp knows that they will read at most k_i messages. Suppose Monocarp pins t messages; if $t \le k_i$, then the i-th student will read all the pinned messages; but if $t > k_i$, the i-th student will choose exactly k_i random pinned messages (all possible subsets of pinned messages of size k_i are equiprobable) and read only the chosen messages.

Monocarp wants to maximize the expected number of students that read their respective messages (i.e. the number of such indices i that student i reads the message m_i). Help him to choose how many (and which) messages should he pin!

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of students in the conference.

Then n lines follow. The i-th line contains two integers m_i and k_i ($1 \le m_i \le 2 \cdot 10^5$; $1 \le k_i \le 20$) — the index of the message which Monocarp wants the i-th student to read and the maximum number of messages the i-th student will read, respectively.

Output

In the first line, print one integer t ($1 \le t \le 2 \cdot 10^5$) — the number of messages Monocarp should pin. In the second line, print t **distinct** integers c_1, c_2, …, c_t ($1 \le c_i \le 2 \cdot 10^5$) — the indices of the messages Monocarp should pin. The messages can be listed in any order.

If there are multiple answers, print any of them.

Examples

standard input	standard output

3
10 1
10 2
5 2
	2
	5 10

3
10 1
5 2
10 1
	1
	10

4
1 1
2 2
3 3
4 4
	3
	2 3 4

3
13 2
42 2
37 2
	3
	42 13 37

Note

Let's consider the examples from the statement.

1. In the first example, Monocarp pins the messages 5 and 10.

   - if the first student reads the message 5, the second student reads the messages 5 and 10, and the third student reads the messages 5 and 10, the number of students which have read their respective messages will be 2;
   - if the first student reads the message 10, the second student reads the messages 5 and 10, and the third student reads the messages 5 and 10, the number of students which have read their respective messages will be 3.

   So, the expected number of students which will read their respective messages is $\frac{5}{2}$.

2. In the second example, Monocarp pins the message 10.

   - if the first student reads the message 10, the second student reads the message 10, and the third student reads the message 10, the number of students which have read their respective messages will be 2.

   So, the expected number of students which will read their respective messages is 2. If Monocarp had pinned both messages 5 and 10, the expected number of students which read their respective messages would have been 2 as well.

3. In the third example, the expected number of students which will read their respective messages is $\frac{8}{3}$.

4. In the fourth example, the expected number of students which will read their respective messages is 2.

Problem I. Tetris

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	64 megabytes

Monocarp is playing a modified version of Tetris game.

The game field is a rectangle divided into square cells. The rectangle contains k rows (numbered 1 through k from bottom to top) and $10^9 + 1$ columns (numbered 0 through 10^9 from left to right). Initially, all cells are empty.

Initially, Monocarp's score in the game is 0. To increase it, Monocarp can drop tetris pieces onto the game field. Each tetris piece is a rectangle with a certain width and a height equal to 1. When the i-th piece appears on the game field, it initially occupies the cells from l_i to r_i (inclusive) in the k-th row of the field. After that, the piece starts falling: it moves one row down (i. e. stops occupying the cells in the current row and starts occupying the cells in the row right below the current one) until it either reaches the row 1, or at least one of the cells directly below the piece is already occupied. Formally, if the piece is currently occupying the cells $[l_i, r_i]$ in the j-th row, it stops if and only if one of the following conditions is met:

- $j = 1$;
- at least one of the cells from l_i-th to r_i-th in the $(j - 1)$-th row is already occupied.

Each tetris piece falls vertically downward, Monocarp cannot force them to move left or right (as it is possible in regular Tetris). While a tetris piece is falling, Monocarp cannot drop any new tetris pieces, he must wait until the current piece stops falling.

Monocarp can drop up to n tetris pieces onto the game field. If the i-th piece is dropped, it initially occupies the cells from l_i to r_i in the k-th row **(and if at least one of those cells is already occupied, the piece cannot be dropped)**; and dropping it increases Monocarp's score by c_i. Monocarp can drop any tetris pieces **in any order**, but each tetris piece can be dropped at most once.

What is the maximum possible score Monocarp can achieve?

Input

The first line contains two integers n and k ($1 \le n \le 5000$; $1 \le k \le \min(n, 10)$).

Then n lines follow, the i-th line contains three integers l_i, r_i and c_i ($0 \le l_i \le r_i \le 10^9$; $0 \le c_i \le 10^9$) — the description of the i-th tetris piece.

Output

Print one integer — the maximum possible score Monocarp can achieve.

Examples

standard input	standard output

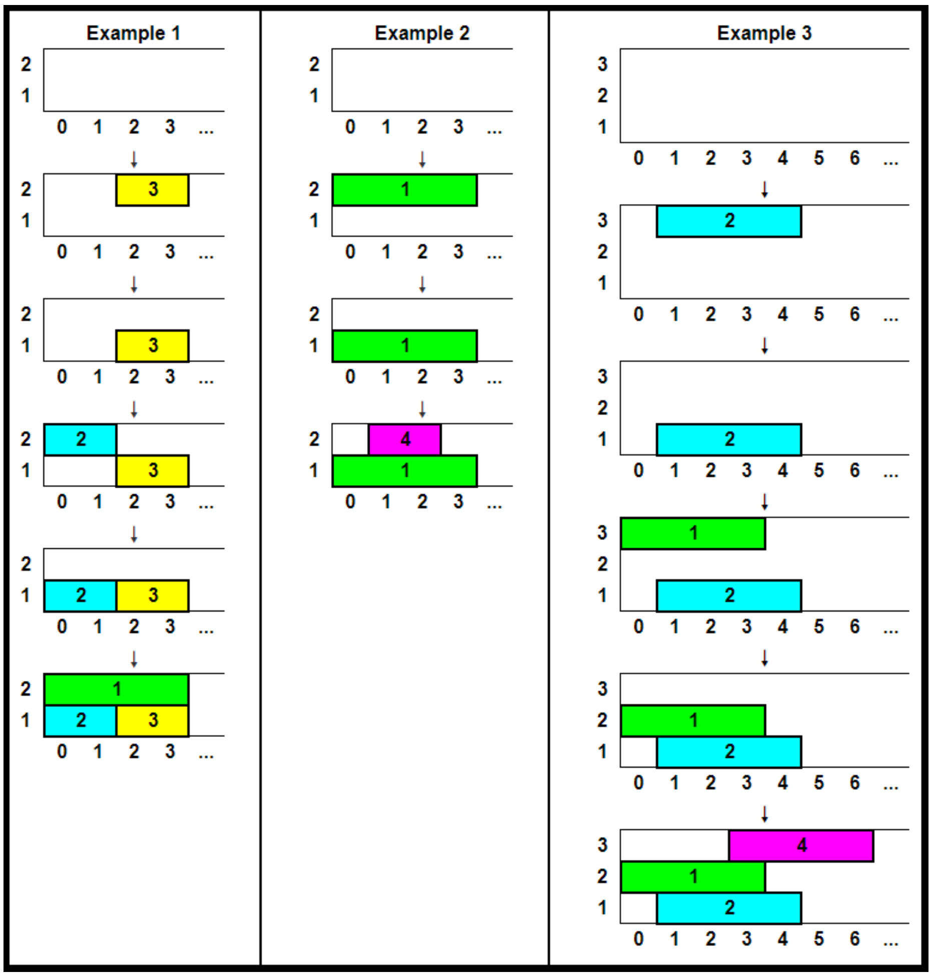
4 2
0 3 30
0 1 5
2 3 10
1 2 14
	45

4 2
0 3 30
0 1 5
2 3 10
1 2 16
	46

4 3
0 3 10
1 4 7
2 5 3
3 6 20
	37

Note

You can see the explanations for the examples in the following picture:



Problem J. Bongcloud Opening

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

Chess Grandmaster Hikaru Nakamura often plays chess on his favorite website. When a game starts, sometimes Hikaru feels like playing a normal game of chess. But in some cases, to make things a bit interesting and tricky, he plays the Bongcloud opening, which is an atrociously bad way to start a game of chess, but may throw some of his opponents off their game plans.

Today, Hikaru is going to play n games of chess. Initially, his rating is x. If he defeats his opponent in the i-th game, his rating goes up by a_i points; if he loses to the i-th opponent, his rating goes down by b_i points (we assume that draws won't happen at all). The order of Hikaru's opponents cannot be changed. For each opponent, Hikaru knows the result of the game with him even before all the games start, though this result may be different if Hikaru plays Bongcloud instead of a normal game of chess. Formally, for the i-th opponent, two integers c_i and d_i are given: $c_i = 1$ means Hikaru defeats his opponent in a normal game of chess, $c_i = 0$ means he loses in a normal game; the number d_i represents the result of the game if Hikaru plays the Bongcloud opening in a same way.

Now, Hikaru has some specific rules to decide whether to start a normal game or a Bongcloud game:

- if Hikaru's current rating is less than $x - k$, he plays a normal chess game;
- if Hikaru's current rating is greater than $x + k$, he plays with the Bongcloud opening;
- if none of those two conditions apply, Hikaru can choose either way to play a game.

What is the maximum possible rating Hikaru can achieve after all n games?

Input

The first line contains three integers n, k and x ($1 \le n \le 2000$; $0 \le k \le 2$; $0 \le x \le 10^8$).

Then n lines follow. The i-th line contains four integers a_i, b_i, c_i and d_i ($0 \le a_i, b_i \le 10^5$; $0 \le c_i, d_i \le 1$) which describe the i-th opponent.

Output

Print one integer — the maximum rating Hikaru can achieve after all n games.

Examples

standard input	standard output

3 2 10
10 2 1 0
6 1 0 1
20 30 1 0
	27

2 1 15
10 20 0 0
40 50 0 1
	-55

Note

In the first example, Hikaru can obtain the maximum possible rating as follows:

- in the first game, Hikaru should choose the Bongcloud opening, so he loses, and his rating becomes $10 - 2 = 8$;

- in the second game, Hikaru should play normally, so he loses, and his rating becomes $8 - 1 = 7$;
- in the third game, Hikaru's rating is lower than $10 - 2$, so he is forced to play normally; he wins, and his rating becomes $7 + 20 = 27$.

In the second example, Hikaru loses both games: he loses in the first game no matter how he plays, and his rating becomes -5. Then, his rating is lower than $15 - 1$, so he is forced to play a normal game of chess, which he loses as well — and his rating becomes -55.

Problem K. Ice Cream Van

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

Polycarp is a huge fan of ice cream. However, he lives in a small city of Bertown, where it's pretty hard to find new types of ice cream to try out.

Luckily for Polycarp, an ice cream van will be arriving to Bertown every day during the next n days. Each day, the van will be bringing an infinite supply of one type of ice cream. The type of ice cream brought on the i-th day has a taste value of a_i.

Now Polycarp wants to come up with an integer value x (the same for all n days) and divide the types of ice cream into tasty and mediocre — an ice cream type will be considered tasty if its taste value is not less than x, otherwise it's mediocre. He proceeds with the following strategy on the i-th of the next n days:

- if Polycarp has at least one ice cream left from previous days, he doesn't buy any ice cream on that day;
- otherwise, if $a_i < x$ (the ice cream that is being sold during this day is mediocre), then he buys 1 ice cream and eats it on the i-th day;
- otherwise, he buys k_i ice creams and eats one of them on each of the days $i, i+1, \ldots, i+k_i-1$. He doesn't buy any more ice creams during these days. Thus, the next time he will visit the van on the day $i+k_i$, if that day doesn't exceed n. Note that $i+k_i$ might exceed n — in that case, Polycarp still eats all k_i ice creams he has already bought.

Polycarp estimates his happiness as the total taste value of the ice creams he eats. In particular, if he eats k_i ice creams of the i-th type, then its taste value a_i gets added to the happiness k_i times.

What is the maximum happiness Polycarp can achieve by choosing the value of x?

Input

The first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of days the van arrives.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the taste value of the ice cream on the i-th day.

The third line contains n integers $k_1, k_2, \ldots, k_n$ ($1 \le k_i \le n$) — the number of ice creams Polycarp wants to buy on the i-th day if the ice cream is tasty.

Output

Print a single integer — the maximum happiness Polycarp can achieve by choosing the value of x.

Examples

standard input	standard output

5
7 3 5 9 3
2 5 5 1 3
	39

4
4 2 1 3
2 2 2 2
	15

4
4 2 1 3
1 1 1 1
	10

Problem L. Smash the Trash

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Vitaly is the mayor of Bertown. He prepares for the new mayor election. As a part of his election campaign, he states that he will make Bertown clean again.

There's not that much time till the election starts, so Vitaly's PR manager Arkady decided to clean only the trash resided in the central street of the city. Namely, there are n dirty places on the central street, numbered from 1 to n from west to east. Arkady wants to dispose of the trash in the i-th place during the i-th day.

As a true patriot of Bertown and the only person in Vitaly's team who is good in arithmetic, you were asked to calculate the minimal number of people to be involved in the cleaning process. This number is selected once before the process starts and stays the same during all n days.

After talking to a cleaning expert, you realized that one worker can perform one of three actions during day i:

- dispose of 1 kilogram of trash in the place i;
- move exactly 2 kilograms of trash from the place i to the place $i + 1$ (of course, this is impossible if $i = n$);
- do nothing.

The street is considered clean if there's no trash in all places at the end of the n-th day. Thus, Arkady has to hire multiple people, so that they can work simultaneously to clean the street.

Help Arkady to calculate the minimal number of people to be involved in the cleaning process.

Input

The first line contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of dirty places in the central street.

The second line contains n integers $t_1, t_2, \ldots, t_n$ ($0 \le t_i \le 2 \cdot 10^5$) — the amount of trash in the i-th place, measured in kilograms.

Output

Print one integer — the minimal number of people to be involved in the cleaning process.

Examples

standard input	standard output

`3`
`5 1 2`
	`4`

`4`
`8 1 2 7`
	`7`

Note

In the first example, Arkady can clean the street by hiring 4 people:

- During the first day, 3 kilograms of trash should be disposed of by 3 people, while 2 kilograms of trash should be moved to the next dirty place by one person.

- During the second day, you have to deal with 3 kilograms of trash in the place 2 using 4 people. It can be easily done if three people dispose of the trash, and one person does nothing.
- During the third day, 4 workers need to dispose of 2 kilograms of trash.

The same thing can't be done by hiring 3 people since otherwise you'll have to move 4 kilograms of trash from place 1, then move 4 kilograms again from place 2 and then face the situation with 6 kilograms of trash in the third and last place, and 3 people cannot dispose of all 6 kilograms.

In the second example, the answer can't be less than 7, since there are 7 kilograms of trash in the last place, and Arkady has to hire at least 7 people to dispose of it.

Problem M. Distance

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

Let's denote the Manhattan distance between two points p_1 (with coordinates (x_1, y_1)) and p_2 (with coordinates (x_2, y_2)) as $d(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$. For example, the distance between two points with coordinates $(1, 3)$ and $(4, 2)$ is $|1 - 4| + |3 - 2| = 4$.

You are given two points, A and B. The point A has coordinates $(0, 0)$, the point B has coordinates (x, y).

Your goal is to find a point C such that:

- both coordinates of C are non-negative integers;
- $d(A, C) = \frac{d(A, B)}{2}$ (without any rounding);
- $d(B, C) = \frac{d(A, B)}{2}$ (without any rounding).

Find any point C that meets these constraints, or report that no such point exists.

Input

The first line contains one integer t ($1 \le t \le 3000$) — the number of test cases.

Each test case consists of one line containing two integers x and y ($0 \le x, y \le 50$) — the coordinates of the point B.

Output

For each test case, print the answer on a separate line as follows:

- if it is impossible to find a point C meeting the constraints, print "`-1 -1`" (without quotes);
- otherwise, print two non-negative integers not exceeding 10^6 — the coordinates of point C meeting the constraints. If there are multiple answers, print any of them. It can be shown that if any such point exists, it's possible to find a point with coordinates not exceeding 10^6 that meets the constraints.

Example

standard input	standard output

10
49 3
2 50
13 0
0 41
42 0
0 36
13 37
42 16
42 13
0 0
	23 3
	1 25
	-1 -1
	-1 -1
	21 0
	0 18
	13 12
	25 4
	-1 -1
	0 0

Note

Explanations for some of the test cases from the example:

- In the first test case, the point B has coordinates $(49, 3)$. If the point C has coordinates $(23, 3)$, then the distance from A to B is $|49 - 0| + |3 - 0| = 52$, the distance from A to C is $|23 - 0| + |3 - 0| = 26$, and the distance from B to C is $|23 - 49| + |3 - 3| = 26$.
- In the second test case, the point B has coordinates $(2, 50)$. If the point C has coordinates $(1, 25)$, then the distance from A to B is $|2 - 0| + |50 - 0| = 52$, the distance from A to C is $|1 - 0| + |25 - 0| = 26$, and the distance from B to C is $|1 - 2| + |25 - 50| = 26$.
- In the third and the fourth test cases, it can be shown that no point with integer coordinates meets the constraints.
- In the fifth test case, the point B has coordinates $(42, 0)$. If the point C has coordinates $(21, 0)$, then the distance from A to B is $|42 - 0| + |0 - 0| = 42$, the distance from A to C is $|21 - 0| + |0 - 0| = 21$, and the distance from B to C is $|21 - 42| + |0 - 0| = 21$.

Problem N. Haiku

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

A *haiku* is a Japanese form of poetry. All haiku poems follow a distinct pattern: they consist of three phrases — the first phrase has 5 syllables, the second phrase has 7 syllables and the third phrase has 5 syllables.

Phonetic nuances aside, let's consider each vowel in a phrase a single syllable. Thus, a phrase contains as many syllables as vowels.

From that, there arises a classic argument of what is considered a vowel. Letters `A`, `E`, `I`, `O`, `U` are vowels for sure (regardless of being lowercase or uppercase). Letter `Y`, however... Let's say that it can be both vowel or consonant, and you are free to decide which one you want it to be each time it appears in the phrase. All other letters are consonants.

You want to check if the given three phrases can fit into the haiku format. That is if it's possible to call every letter `Y` in them a vowel or a consonant in such a way that the number of syllables in the phrases is $5 - 7 - 5$.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of testcases.

Then the descriptions of t testcases follow.

The first line of the testcase contains three integers n_1, n_2, n_3 ($1 \le n_i \le 50$) — the number of words in the i-th phrase. A word is a sequence of lowercase and uppercase Latin letters.

The i-th of the next three lines contains the i-th phrase: n_i words, separated by a single space. The first and the last characters of the phrase are always letters. The length of each phrase doesn't exceed 100.

Output

For each testcase print "`YES`" if it's possible to call every letter `Y` in the given three phrases a vowel or a consonant in such a way that the number of syllables in the phrases is $5 - 7 - 5$. Otherwise, print "`NO`".

Example

standard input

```
7
3 4 4
Rainy and cold
On the programming contest
Think of the tree
4 5 3
That reminds me of
How I became a cf
Master on Monday
4 6 5
Buy it use it
Break it fix it trash it
Change it mail upgrade it
2 5 3
Today things
Well known in China are
Well known globally
2 4 3
Seems interesting
I will definitely try
To implement it
2 3 4
U lukomorya
dub zelenyy Zlataya
cep na dube tom
3 9 3
I dont understand
why why why why oh why why why why
thats a haiku
```

standard output

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