Problem A. Yet Another Problem with Strings

Input file:	`standard input`
Output file:	`standard output`
Time limit:	8 seconds
Memory limit:	256 megabytes

You are given an array with n strings $S_0, S_1, \ldots, S_{n-1}$. You must process q queries, numbered 0 through $q-1$. All strings in this problem are non-empty and consist of lowercase English letters.

Let's create an integer variable `LAST_YES` to decipher the input. As you process the queries in the given order, `LAST_YES` should be equal to the index of the last query that was of the first type and had an answer `YES`. `LAST_YES` is initially equal to 0. Note that `LAST_YES` can't be changed in the first query because the index of the first query is 0 (note again that queries are numbered from 0 to $q-1$).

There are two types of queries.

The first type is of ciphered form "`1 t`", where t is a string of lowercase English characters. Add `LAST_YES` to each character in t (modulo 26) to get its deciphered value. For example, for a string `t = "cdxyyz"` with `LAST_YES` equal to 2, the deciphered string is `"efzaab"`. Then you should check whether at least one string S_i is a substring of the deciphered string. In a single line print `YES` or `NO`.

The second type is of ciphered form "`2 i alpha`" where i and $alpha$ are integers. You should add a letter $(alpha + LAST\_YES)\%26$ at the end of $S_{(i+LAST\_YES)\%n}$. Here we treat letters `'a'-'z'` as numbers $0-25$. For example, for $alpha = 23$ ans $LAST\_YES = 2604$, you get $(23+2604)\%26 = 1$, so you should add a letter `'b'` at the end of $S_{(i+LAST\_YES)\%n}$.

Input

The first line of the input contains two integers n and q $(1 \le n, q, \le 200\,000)$ — the size of the array S and the number of queries, respectively.

The next n lines contain strings $S_0, S_1, \ldots, S_{n-1}$, one string per line. All strings are nonempty and consist of lowercase English letters. The total length of all n strings doesn't exceed 200 000.

The next q lines contain queries. Each query is of form "`1 t`" or "`2 i alpha`".

For the first type of query, t is a nonempty string of lowercase English letters. The total length of all t doesn't exceed 200 000.

For the second type, i and $alpha$ are integers, and $0 \le i < n, 0 \le alpha < 26$.

Output

For each query of the first type, print `YES` if there exists a string S_i that is a substring of the deciphered string t. Otherwise print `NO` (without the quotes).

Examples

standard input	standard output

`5 7`
`aba`
`broadway`
`test`
`why`
`i`
`1 tst`
`1 text`
`2 1 2`
`1 caaabaaac`
`1 qbpqfkd`
`2 4 0`
`1 wdwdsdubaa`
	`NO`
	`NO`
	`YES`
	`YES`
	`NO`

Note

The answer is `NO` for the queries 0 and 1 because no initial S_i is a substring of `"tst"` or `"text"`.

In the query 2 a string S_1 is changed from `"broadway"` into `"broadwayc"`.

In the query 3 the answer is `YES` because a string S_0 (`"aba"`) is a substring of `"caaabaaac"`. `LAST_YES` is equal to 3 now.

The query 4 asks about deciphered string `"testing"` for which a string S_4 (`"i"`) is a substring. $LAST\_YES$ is 4 now.

The query 5 asks to add a letter `'e'` (because $0 + LAST\_YES = 4$) at the end of S_3 (because $(4 + LAST\_YES)\%n = 8\%5 = 3$). So, we change `"why"` to `"whye"`.

In the last query a deciphered string t is `"ahahwhyfee"` after the deciphering. None of S_i is its substring, so the answer is `NO`.

Problem B. Pen Pineapple Apple Pen

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

PPAP, which stands for Pen Pineapple Apple Pen, is an unusual song and dance that went viral in the Internet recently. It's about merging a pen with either an apple or a pineapple, and then merging those together into more complicated structures.

A *structure* is either one of mentioned items (a pen, an apple or a pineapple) or a result of combining two structures. In the former case we say a structure is *basic*.

The size of a basic structure is 1. The size of any other structure is equal to the sum of sizes of two structures merged into this one.

Two structures can be merged if one of the following conditions holds true:

1. Both structures are basic and exactly one of them is a pen.
2. None of two structures are basic and their sizes are equal.

A structure can be represented by a string of length equal to the size of the structure. A basic structure is represented by `"A"` (if it's an apple) or by `"P"` (if it's a pen or a pineapple). When a structure represented by a string w_1 and a structure represented by a string w_2 are merged together, they create a structure represented by a string w_1w_2 (strings are concatenated). The order of two merged structures matters because w_1w_2 may be different that w_2w_1.

You are given T test cases, each with one string s_i consisting of characters `'A'` and `'P'`. Your task is to check whether it's possible to get a structure represented by such a string. Print `"YES"` or `"NO"` in a separate line, without the quotes.

Input

The first line of the input contains an integer T ($1 \le T \le 10$) — the number of test cases.

Each of the next T lines contains one string s_i ($1 \le |s_i| \le 100$), describing one test case. A string s_i consists of characters `'A'` and `'P'` only.

Output

For each test case in a separate line print `"YES"` (without the quotes) if it's possible to get a structure represented by the given string, and `"NO"` otherwise (without the quotes).

Example

standard input	standard output
4 PPAP A AAAA PPAPP	YES YES NO NO

Note

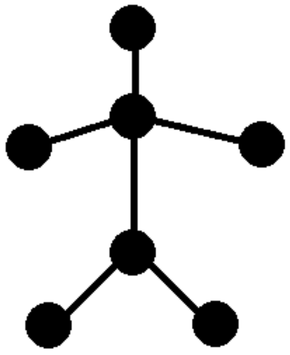
In the first test case, one valid way is to merge a pen and a pineapple to get `"PP"`, then merge an apple and a pen to get `"AP"`, and finally merge two created structured in the order (`"PP"`, `"AP"`). The final structure is represented by `"PPAP"`, as required.

Problem C. Stickmen

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

Limak is a little bear who loves to play with graphs. Recently, he defined a new structure in a graph and called it a *stickman*.

A stickman is a set of 7 different vertices and 6 edges. Vertices should be connected by edges as in the image below.



You can see here two legs, two arms and a head. Formally, one vertex (the one between arms) must be connected to four other vertices and one of these four must be connected to two more vertices.

Limak has an undirected graph with n vertices and m edges, without loops and multiple edges. He asks you a favor. Could you count how many stickmen there are in Limak's graph, modulo $10^9 + 7$?

A stickman is defined by a set of 7 vertices and 6 edges. Two stickmen are different if and only if the set of vertices isn't the same or the set of edges isn't the same. The order of vertices or edges doesn't matter (so you shouldn't distinguish legs for example).

Input

The first line of the input contains two integers n and m ($7 \le n \le 200$, $0 \le m \le \frac{n \cdot (n-1)}{2}$) — the number of vertices and the number of edges. Vertices are numbered 1 through n.

Each of the next m lines contains two integers a_i and b_i ($1 \le a_i, b_i$, $a_i \ne b_i$) — indices of two vertices connected with an edge. Any unordered pair of vertices will appear at most once.

Output

Print the number of stickmen in the given graph, modulo $10^9 + 7$.

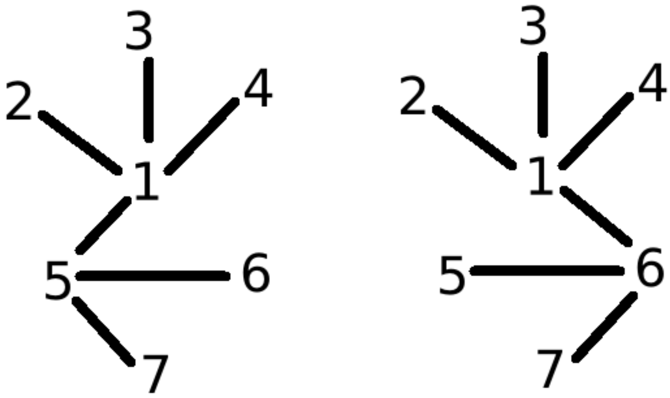
Example

standard input	standard output

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

Note

There are two different stickmen in the sample, showed in the drawing below. Since there are only 7 vertices in the graph, the set of vertices of each stickman must contain all vertices. The two stickmen don't have the same set of edges though and this is why they are different at all.



Problem D. Strange Queries

Input file: stdin
Output file: stdout
Time limit: 3 seconds
Memory limit: 256 megabytes

You are given an array with n integers $a_1, a_2, \ldots, a_n$, and q queries to answer.

Each query consists of four integers l_1, r_1, l_2 and r_2. Your task is to count the number of pairs of indices (i, j) satisfying the following conditions:

1. $a_i = a_j$
2. $l_1 \le i \le r_1$
3. $l_2 \le j \le r_2$

Input

The first line of the input contains an integer n ($1 \le n \le 50\,000$) — the size of the given array.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$).

The third line contains an integer q ($1 \le q \le 50\,000$) — the number of queries.

Each of the next q lines contains four integers l_1, r_1, l_2, r_2 ($1 \le l_1 \le r_1 \le n$, $1 \le l_2 \le r_2 \le n$), describing one query.

Output

For each query count the number of sought pairs of indices (i, j) and print it in a separate line.

Examples

stdin	stdout

7
1 5 2 1 7 2 2
8
1 3 4 5
2 3 5 7
1 4 3 7
2 6 4 7
1 6 2 5
3 3 6 7
4 5 1 4
2 3 4 5
	1
	2
	5
	6
	6
	2
	2
	0

Problem E. Bravebeart

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

Would you want to fight against bears riding horses? Me neither.

Limak is a grizzly bear. He is a general of the dreadful army of Bearland. The most important part of an army is the cavalry of course.

The cavalry of Bearland consists of n warriors and n horses, both numbered 1 through n. Limak knows the strength of each warrior $w_1, w_2, \ldots, w_n$ and of each horse $h_1, h_2, \ldots, h_n$.

A warrior together with his horse is called a unit. The strength of a unit is equal to the multiplied strengths of a warrior and a horse.

General Limak must assign all horses to warriors, one horse per warrior. The cavalry will consist of n units then.

The first warrior (the one with the strength w_1) is called Bravebeart. He is always the first to charge the enemy. Limak decided that Bravebeart deserves some respect and his unit must be the strongest one, with no ties allowed. But is it possible?

Help Limak and check whether there is an assignment of horses to warriors for which the Bravebeart's unit is strictly stronger than any other unit. Print "YES" or "NO" (without the quotes).

Input

You are given multiple test cases.

The first line of the input contains one integer T ($1 \le T \le 50$), denoting the number of test cases.

For each test case the first line contains one integer n ($2 \le n \le 42$).

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($1 \le w_i \le 1000$), denoting strengths of warriors. The first number is the strength of Bravebeart.

The third line contains n integers $h_1, h_2, \ldots, h_n$ ($1 \le h_i \le 1000$), denoting strengths of horses.

Output

For each test case find the answer and print it in a separate line.

Print "YES" (without the quotes) if there is an assignment where the strength of the Bravebeart's unit is strictly greater than strength of any other unit. Otherwise print "NO" (without the quotes).

Example

standard input	standard output

2
6
12 9 7 1 20 10
3 6 4 7 5 5
4
5 17 10 1
200 400 800 600
	YES
	NO

Note

In the first test case Bravebeart can take a horse of strength 6 to get the unit strength $12 \cdot 6 = 72$.

In one way of assigning other horses to warriors the pairs (strength of warrior, strength of horse) are: $(9, 7)$, $(7, 5)$, $(1, 5)$, $(20, 3)$, $(10, 4)$. Units strengths would be 63, 35, 5, 60 and 40, respectively. Indeed, the Bravebeart's unit is stronger than any other unit then.

In the second test case it's impossible to assign horses to warriors so that Bravebeart's unit is stronger than any other one.

Problem F. GukiZ Height

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

GukiZ loves hiking in the mountains. He starts his hike at the height 0 and he has some goal to reach, initially equal to h.

The mountains are described by a sequence of integers $A_0, A_1, \ldots, A_{n-1}$. In the first day GukiZ will change his height by A_0, in the second day by A_1, and so on. Mountains are a regular structure so the pattern repeats after the first n days. In general, in the i-th day GukiZ will change his height by $A_{(i-1)\%n}$.

Additionally, GukiZ will become more and more tired and in the i-th day his goal will decrease by i. For example, after first three days his goal will be equal to $h - 1 - 2 - 3 = h - 6$.

Note that A may contain negative elements, what represents going down from some hill. Moreover, GukiZ's height may become negative, and so does his goal!

You can assume that both GukiZ's height and goal change at the same moment (immediately and simultaneously) in the middle of a day.

Once GukiZ is at the height not less than his goal, he ends his hike. Could you calculate the number of days in his hike?

Input

The first line of the input contains two integers n and h ($1 \le n \le 10^5$, $1 \le h \le 10^9$) — the length of the array A and the initial goal, respectively.

The second line contains n integers $A_0, A_1, \ldots, A_{n-1}$ ($-10^9 \le A_i \le 10^9$).

Output

In a single line print the number of days in the GukiZ's hike.

It can be proved that for any valid input the answer exists.

Examples

standard input	standard output
3 45 7 -4 5	7
1 1 0	1

Note

GukiZ starts at height 0 with goal 45. We can describe his hike as follows:

1. After the first day he is at height 7 and his goal is 44.
2. Height 3, goal 42.
3. Height 8, goal 39.
4. Height 15, goal 35.
5. Height 11, goal 30.
6. Height 16, goal 24.
7. Height 23, goal 17.

After the 7-th day Gukiz's height is not less than his goal so he ends his hike.

Problem G. LCM-er

Input file: stdin
Output file: stdout
Time limit: 0.5 секунд
Memory limit: 256 мегабайт

You are given four integers n, a, b and x. Your task is to count modulo $10^9 + 7$ the number of sequences of integers A satisfying the following conditions:

1. A should contain exactly n elements.
2. $a \le A_1 \le A_2 \le \ldots \le A_n \le b$ (the sequence is non-decreasing and all integers are between a and b, inclusive).
3. The least common multiple of all numbers in A should be divisible by x.

Input

The only line of the input contains four integers n, a, b and x ($1 \le n \le 100$, $1 \le a, b, x \le 10^9$, $a \le b$).

Output

Count sequences of integers satisfying the given conditions. Print the answer modulo $10^9 + 7$.

Examples

stdin	stdout
2 1 7 6	9
1 1 50 7	7

In the first sample there are 9 valid sequences: $(1, 6)$, $(2, 3)$, $(2, 6)$, $(3, 4)$, $(3, 6)$, $(4, 6)$, $(5, 6)$, $(6, 6)$, $(6, 7)$.

Problem H. Precise Shopping

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

When Baklazan goes shopping, he likes calculating how much he should pay and prepare that amount exactly. There are n coins in his wallet, numbered 1 through n. Coin i has denomination d_i. Also, each coin is somewhat interesting to Baklazan; coin i has the interestingness of a_i.

This time, Baklazan forgot the first step — he doesn't know the exact amount of money to pay. He only remembers that it won't exceed b. Therefore, he'd like to prepare a subset S of his coins such that there's a subset of S with the sum of denominations equal to k for every k between 1 and b, inclusive.

Of course, it's better not to give away interesting coins. Therefore, if there are multiple ways to choose S, he'd like to minimise the sum of interestingnesses of coins in S. Note that it's not necessary to minimise the size of S.

Input

The first line of the input contains two integers n and b ($1 \le n \le 42$, $1 \le b \le 10^9$).

The i-th of the next n lines contains two integers d_i and a_i ($1 \le d_i, a_i \le 10^9$) — the denomination and the interestingness of the i-th coin.

Output

If it's impossible to choose a suitable subset S, print "`-1`" (without the quotes) on a single line.

Otherwise, print two lines. On the first line, print one integer denoting the size of your chosen subset S. On the second line, print the space-separated indices of chosen coins. You may print the indices in any order. If there are multiple ways to choose an optimal S, you may print any of them.

Examples

standard input	standard output

7 11
4 500
5 600
6 700
1 200
2 300
3 400
2 9001
	4
	4 5 6 2

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

Problem I. Prime Moving

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

The author of this problem hopes that you already know the pig called Benny.

Recently she started learning numbers. The first type of numbers she has learnt are primes, i.e. positive integers with exactly two divisors. First few primes are 2, 3, 5, 7 and 11.

As a homework Benny was given two primes a and b. She has to transform a into b using addition and subtraction.

Since Benny knows only primes, she might add and subtract only primes. Moreover, all the partial results also have to be primes.

One move is either adding or subtracting a prime to/from the current number. A way of transforming a into b is called *optimal* if it doesn't use more moves than any other way. Your task is to find the only optimal way, if exactly one exists.

If it's impossible to transform a into b, print `"Unlucky Benny"` (without the quotes). If there are multiple optimal ways, print `"Poor Benny"` (without the quotes). Otherwise output the transformation with all partial results. See the output format below.

Input

The only line of the input contains two integers a and b ($2 \le a < b \le 10^15$). Both a and b are primes.

Output

If the transformation is impossible, print `"Unlucky Benny"`.

If there are many optimal ways, print `"Poor Benny"`.

If there is exactly one optimal way to transform a into b, describe the process in the following format. Let $x_1, x_2, \ldots, x_k$ denote values of Benny's number in the order. So $x_1 = a$, $x_k = b$ and Benny changes x_i to x_{i+1} in the i-th move. In a single line print k numbers separated with two characters `>"`, without the quotes and spaces.

Example

standard input	standard output
5 7	5->7

Note

One move is enough in the sample.

If the only optimal way was to change 5 to 20, then change 20 to 10, and then finally change 10 to 7, the output would be `5->20->10->7`. Though, such a way is not only non-optimal but also it isn't valid. In the first move (from 5 to 20) we try to add 15 what is not prime. Also, a new number 20 isn't prime. This note is only to show you the correct format.

Problem J. Valentina and the Gift Tree

Input file: `standard input`
Output file: `standard output`
Time limit: 6 seconds
Memory limit: 256 megabytes

The Valentina's birthday is coming soon! Her parents love her so much that they are preparing some gifts to give her. Valentina is cute and smart and her parents are planning to set an interesting task for her.

They prepared a tree (a connected graph without cycles) with one gift in each vertex. Vertices are numbered 1 through n and the i-th of them contains a gift with value g_i (a value describing Valentina's happiness if she gets a gift from this vertex). All gifts are wrapped so Valentina doesn't know their values.

Note that g_i may be negative and it would mean that Valentina doesn't like the gift (do you remember when your parents gave you clothes or toys not adequate to your interests?).

Let's consider the following process:

1. Valentina chooses two vertices a and b (not necessarily distinct).
2. She unpacks all gifts on the path connecting a and b and thus she sees their values.
3. She chooses some part of the path connecting a and b. The chosen part must be connected and can't be empty (hence, it must be a path).
4. Valentina takes all gifts in the chosen part of the path and her total happiness is equal to the sum of the values of taken gifts.

Valentina is smart and for chosen a and b she will choose a part resulting in the biggest total happiness.

In order to maximize her chance to get a good bunch of gifts, parents allow her to ask q questions, each with two numbers a and b. For each Valentina's question parents will tell her the answer, i.e. the maximum total happiness for chosen a and b.

They noticed that answering Valentina's questions is an interesting problem. They want to know if you are smart enough to correctly answer all questions.

Input

The first line of the input contains one integer n ($2 \le n \le 10^5$) — the size of the tree.

Each of the next $n - 1$ lines contains two integers u_i and v_i ($1 \le u_i, v_i \le n$) — indices of two vertices connected with an edge. The given edges are guaranteed to describe a tree.

The next line contains n integers $g_1, g_2, \ldots g_n$ ($-10^9 \le g_i \le 10^9$).

Then, the questions are given. One line contains an integer q ($1 \le q \le 500\,000$) — the number of questions.

Finally, each of the last q lines contains two integers a_i and b_i ($1 \le a_i, b_i \le n$), describing one question.

Output

For each of the q questions find the maximum happiness of Valentina if she has to choose a non-empty part of the given path. For every question print the answer in a separate line.

Examples

standard input	standard output

6
1 2
1 3
1 4
4 5
4 6
-1 2 2 -2 5 8
6
2 5
2 3
1 5
5 3
1 4
5 6
	5
	3
	5
	5
	-1
	11

Problem K. The World of Trains

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

In the computer world, a train of the length n can be represented as an array A of the same length. Such an array A contains integer elements between 1 and k, inclusive. Each element in A represents the quality of one carriage. You can notice that there are exactly k^n distinct trains.

Mike loves to travel and today he is going to do it with his friends. They want to occupy exactly d consecutive carriages. Of course, they want chosen d carriages to be of equal quality.

Someone saw a train and thus knows an array A. He said that there are exactly t ways for them to choose d consecutive carriages of equal quality. Now, Mike wants to know how many trains satisfy this description. Your task is to count them modulo $10^9 + 7$.

Input

The only line of the input contains four integers n, d, t and k ($1 \le d \le n \le 3333$, $0 \le t \le n - d + 1$, $1 \le k \le 10^9$).

Output

Find the number of distinct trains fitting the given description, and print it modulo $10^9 + 7$.

Examples

standard input	standard output
5 2 1 2	8
3 2 2 5	5
3 2 0 5	80

Note

For the first sample test the only possible arrays are:

1. $1, 1, 2, 1, 2$
2. $1, 2, 1, 1, 2$
3. $1, 2, 1, 2, 2$
4. $1, 2, 2, 1, 2$
5. $2, 1, 1, 2, 1$
6. $2, 1, 2, 1, 1$
7. $2, 1, 2, 2, 1$
8. $2, 2, 1, 2, 1$