A. Hotelier

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Amugae has a hotel consisting of 10 rooms. The rooms are numbered from 0 to 9 from left to right.

The hotel has two entrances — one from the left end, and another from the right end. When a customer arrives to the hotel through the left entrance, they are assigned to an empty room closest to the left entrance. Similarly, when a customer arrives at the hotel through the right entrance, they are assigned to an empty room closest to the right entrance.

One day, Amugae lost the room assignment list. Thankfully Amugae's memory is perfect, and he remembers all of the customers: when a customer arrived, from which entrance, and when they left the hotel. Initially the hotel was empty. Write a program that recovers the room assignment list from Amugae's memory.

Input

The first line consists of an integer n ($1 \le n \le 10^5$), the number of events in Amugae's memory.

The second line consists of a string of length n describing the events in chronological order. Each character represents:

- 'L': A customer arrives from the left entrance.
- 'R': A customer arrives from the right entrance.
- '0', '1', ..., '9': The customer in room x (0, 1, ..., 9 respectively) leaves.

It is guaranteed that there is at least one empty room when a customer arrives, and there is a customer in the room x when x (0, 1, ..., 9) is given. Also, all the rooms are initially empty.

Output

In the only line, output the hotel room's assignment status, from room 0 to room 9. Represent an empty room as '0', and an occupied room as '1', without spaces.

Examples

input
8 LLRL1RL1
output
1010000011

input
9 L0L0LLRR9
output
1100000010

Note

In the first example, hotel room's assignment status after each action is as follows.

- First of all, all rooms are empty. Assignment status is 0000000000.
- L: a customer arrives to the hotel through the left entrance. Assignment status is 1000000000.
- L: one more customer from the left entrance. Assignment status is 1100000000.
- R: one more customer from the right entrance. Assignment status is 1100000001.
- L: one more customer from the left entrance. Assignment status is 1110000001.
- 1: the customer in room 1 leaves. Assignment status is 1010000001.
- R: one more customer from the right entrance. Assignment status is 1010000011.
- L: one more customer from the left entrance. Assignment status is 1110000011.
- 1: the customer in room 1 leaves. Assignment status is 1010000011.

So after all, hotel room's final assignment status is 1010000011.

In the second example, hotel room's assignment status after each action is as follows.

- L: a customer arrives to the hotel through the left entrance. Assignment status is 1000000000.
- 0: the customer in room 0 leaves. Assignment status is 0000000000.
- L: a customer arrives to the hotel through the left entrance. Assignment status is 1000000000 again.

- 0: the customer in room 0 leaves. Assignment status is `0000000000`.
- L: a customer arrives to the hotel through the left entrance. Assignment status is `1000000000`.
- L: one more customer from the left entrance. Assignment status is `1100000000`.
- R: one more customer from the right entrance. Assignment status is `1100000001`.
- R: one more customer from the right entrance. Assignment status is `1100000011`.
- 9: the customer in room 9 leaves. Assignment status is `1100000010`.

So after all, hotel room's final assignment status is `1100000010`.

B. Block Adventure

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Gildong is playing a video game called *Block Adventure*. In Block Adventure, there are n columns of blocks in a row, and the columns are numbered from 1 to n. All blocks have equal heights. The height of the i-th column is represented as h_i, which is the number of blocks stacked in the i-th column.

Gildong plays the game as a character that can stand only on the top of the columns. At the beginning, the character is standing on the top of the 1-st column. The goal of the game is to move the character to the top of the n-th column.

The character also has a bag that can hold infinitely many blocks. When the character is on the top of the i-th column, Gildong can take one of the following three actions as many times as he wants:

- if there is at least one block on the column, remove one block from the top of the i-th column and put it in the bag;
- if there is at least one block in the bag, take one block out of the bag and place it on the top of the i-th column;
- if $i < n$ and $|h_i - h_{i+1}| \le k$, move the character to the top of the $i+1$-st column. k is a non-negative integer given at the beginning of the game. Note that it is only possible to move to the **next** column.

In actions of the first two types the character remains in the i-th column, and the value h_i changes.

The character initially has m blocks in the bag. Gildong wants to know if it is possible to win the game. Help Gildong find the answer to his question.

Input
Each test contains one or more test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The first line of each test case contains three integers n, m, and k ($1 \le n \le 100$, $0 \le m \le 10^6$, $0 \le k \le 10^6$) — the number of columns in the game, the number of blocks in the character's bag at the beginning, and the non-negative integer k described in the statement.

The second line of each test case contains n integers. The i-th integer is h_i ($0 \le h_i \le 10^6$), the initial height of the i-th column.

Output
For each test case, print "YES" if it is possible to win the game. Otherwise, print "NO".

You can print each letter in any case (upper or lower).

Example

input
5 3 0 1 4 3 5 3 1 2 1 4 7 4 10 0 10 20 10 20 2 5 5 0 11 1 9 9 99
output
YES NO YES NO YES

Note
In the first case, Gildong can take one block from the 1-st column, move to the 2-nd column, put the block on the 2-nd column, then move to the 3-rd column.

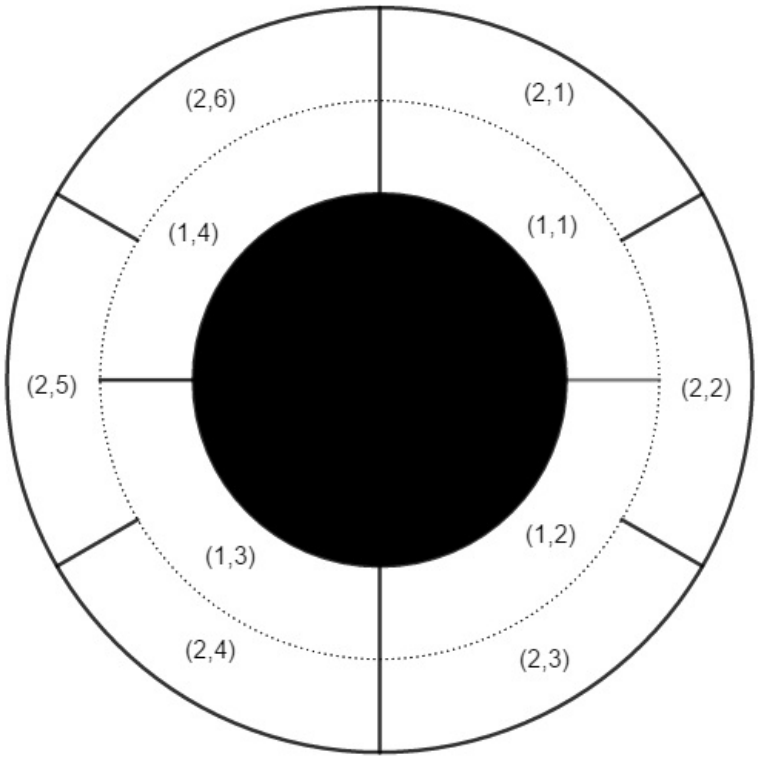
In the second case, Gildong has to put the block in his bag on the 1-st column to get to the 2-nd column. But it is impossible to get to the 3-rd column because $|h_2 - h_3| = 3 > k$ and there is no way to decrease the gap.

In the fifth case, the character is already on the n-th column from the start so the game is won instantly.

C. Round Corridor

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Amugae is in a very large round corridor. The corridor consists of two areas. The inner area is equally divided by n sectors, and the outer area is equally divided by m sectors. A wall exists between each pair of sectors of same area (inner or outer), but there is no wall between the inner area and the outer area. A wall always exists at the 12 o'clock position.



The inner area's sectors are denoted as $(1,1), (1,2), \ldots, (1,n)$ in clockwise direction. The outer area's sectors are denoted as $(2,1), (2,2), \ldots, (2,m)$ in the same manner. For a clear understanding, see the example image above.

Amugae wants to know if he can move from one sector to another sector. He has q questions.

For each question, check if he can move between two given sectors.

Input

The first line contains three integers n, m and q ($1 \le n, m \le 10^{18}$, $1 \le q \le 10^4$) — the number of sectors in the inner area, the number of sectors in the outer area and the number of questions.

Each of the next q lines contains four integers s_x, s_y, e_x, e_y ($1 \le s_x, e_x \le 2$; if $s_x = 1$, then $1 \le s_y \le n$, otherwise $1 \le s_y \le m$; constraints on e_y are similar). Amague wants to know if it is possible to move from sector (s_x, s_y) to sector (e_x, e_y).

Output

For each question, print "YES" if Amugae can move from (s_x, s_y) to (e_x, e_y), and "NO" otherwise.

You can print each letter in any case (upper or lower).

Example

input
4 6 3 1 1 2 3 2 6 1 2 2 6 2 4
output
YES NO YES

Note

Example is shown on the picture in the statement.

D. White Lines

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Gildong has bought a famous painting software *cfpaint*. The working screen of cfpaint is square-shaped consisting of n rows and n columns of square cells. The rows are numbered from 1 to n, from top to bottom, and the columns are numbered from 1 to n, from left to right. The position of a cell at row r and column c is represented as (r, c). There are only two colors for the cells in cfpaint — black and white.

There is a tool named *eraser* in cfpaint. The eraser has an integer size k ($1 \le k \le n$). To use the eraser, Gildong needs to click on a cell (i, j) where $1 \le i, j \le n - k + 1$. When a cell (i, j) is clicked, all of the cells (i', j') where $i \le i' \le i + k - 1$ and $j \le j' \le j + k - 1$ become white. In other words, a square with side equal to k cells and top left corner at (i, j) is colored white.

A *white line* is a row or a column without any black cells.

Gildong has worked with cfpaint for some time, so some of the cells (possibly zero or all) are currently black. He wants to know the maximum number of *white lines* after using the eraser **exactly once**. Help Gildong find the answer to his question.

Input

The first line contains two integers n and k ($1 \le k \le n \le 2000$) — the number of rows and columns, and the size of the eraser.

The next n lines contain n characters each without spaces. The j-th character in the i-th line represents the cell at (i, j). Each character is given as either 'B' representing a black cell, or 'W' representing a white cell.

Output

Print one integer: the maximum number of white lines after using the eraser exactly once.

Examples

input
4 2 BWWW WBBW WBBW WWWB
output
4

input
3 1 BWB WWB BWB
output
2

input
5 3 BWBBB BWBBB BBBBB BBBBB WBBBW
output
2

input
2 2 BW WB
output
4

input
2 1 WW WW
output
4

Note

In the first example, Gildong can click the cell $(2, 2)$, then the working screen becomes:

BWWW
WWWW
WWWW
WWWB

Then there are four white lines — the 2-nd and 3-rd row, and the 2-nd and 3-rd column.

In the second example, clicking the cell $(2, 3)$ makes the 2-nd row a white line.

In the third example, both the 2-nd column and 5-th row become white lines by clicking the cell $(3, 2)$.

E. Compress Words

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Amugae has a sentence consisting of n words. He want to compress this sentence into one word. Amugae doesn't like repetitions, so when he merges two words into one word, he removes the longest prefix of the second word that coincides with a suffix of the first word. For example, he merges "`sample`" and "`please`" into "`samplease`".

Amugae will merge his sentence left to right (i.e. first merge the first two words, then merge the result with the third word and so on). Write a program that prints the compressed word after the merging process ends.

Input

The first line contains an integer n ($1 \le n \le 10^5$), the number of the words in Amugae's sentence.

The second line contains n words separated by single space. Each words is non-empty and consists of uppercase and lowercase English letters and digits ('A', 'B', ..., 'Z', 'a', 'b', ..., 'z', '0', '1', ..., '9'). The total length of the words does not exceed 10^6.

Output

In the only line output the compressed word after the merging process ends as described in the problem.

Examples

input
5 I want to order pizza
output
Iwantorderpizza

input
5 sample please ease in out
output
sampleaseinout

F. Graph Traveler

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

Gildong is experimenting with an interesting machine *Graph Traveler*. In Graph Traveler, there is a directed graph consisting of n vertices numbered from 1 to n. The i-th vertex has m_i outgoing edges that are labeled as $e_i[0]$, $e_i[1]$, ..., $e_i[m_i - 1]$, each representing the destination vertex of the edge. The graph can have multiple edges and self-loops. The i-th vertex also has an integer k_i written on itself.

A *travel* on this graph works as follows.

1. Gildong chooses a vertex to start from, and an integer to start with. Set the variable c to this integer.
2. After arriving at the vertex i, or when Gildong begins the travel at some vertex i, add k_i to c.
3. The next vertex is $e_i[x]$ where x is an integer $0 \le x \le m_i - 1$ satisfying $x \equiv c \pmod{m_i}$. Go to the next vertex and go back to step 2.

It's obvious that a travel never ends, since the 2nd and the 3rd step will be repeated endlessly.

For example, assume that Gildong starts at vertex 1 with $c = 5$, and $m_1 = 2$, $e_1[0] = 1$, $e_1[1] = 2$, $k_1 = -3$. Right after he starts at vertex 1, c becomes 2. Since the only integer x ($0 \le x \le 1$) where $x \equiv c \pmod{m_i}$ is 0, Gildong goes to vertex $e_1[0] = 1$. After arriving at vertex 1 again, c becomes -1. The only integer x satisfying the conditions is 1, so he goes to vertex $e_1[1] = 2$, and so on.

Since Gildong is quite inquisitive, he's going to ask you q queries. He wants to know how many **distinct** vertices will be visited

infinitely many times, if he starts the travel from a certain vertex with a certain value of c. Note that you should **not** count the vertices that will be visited only finite times.

Input

The first line of the input contains an integer n ($1 \leq n \leq 1000$), the number of vertices in the graph.

The second line contains n integers. The i-th integer is k_i ($-10^9 \leq k_i \leq 10^9$), the integer written on the i-th vertex.

Next $2 \cdot n$ lines describe the edges of each vertex. The $(2 \cdot i + 1)$-st line contains an integer m_i ($1 \leq m_i \leq 10$), the number of outgoing edges of the i-th vertex. The $(2 \cdot i + 2)$-nd line contains m_i integers $e_i[0]$, $e_i[1]$, ..., $e_i[m_i - 1]$, each having an integer value between 1 and n, inclusive.

Next line contains an integer q ($1 \leq q \leq 10^5$), the number of queries Gildong wants to ask.

Next q lines contains two integers x and y ($1 \leq x \leq n$, $-10^9 \leq y \leq 10^9$) each, which mean that the start vertex is x and the starting value of c is y.

Output

For each query, print the number of **distinct** vertices that will be visited **infinitely many times**, if Gildong starts at vertex x with starting integer y.

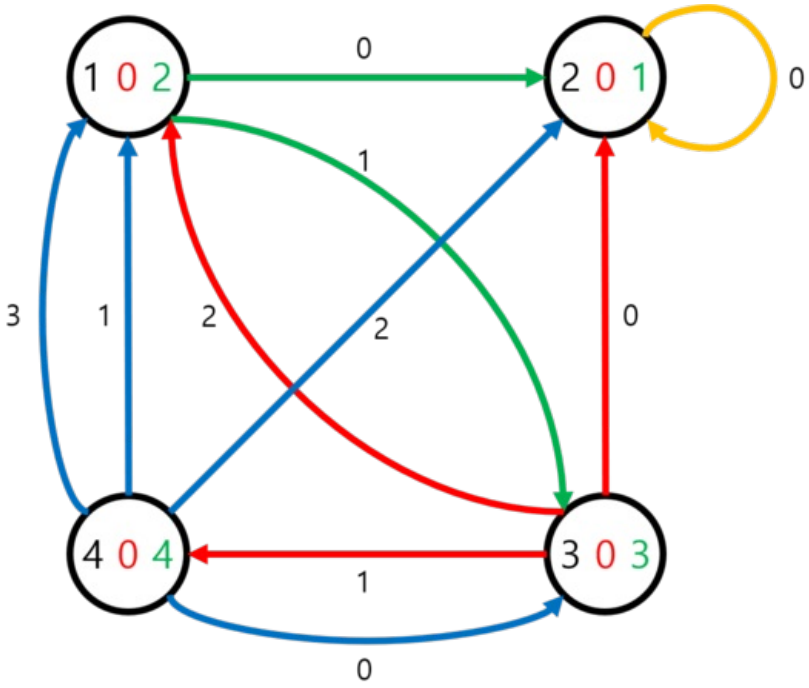
Examples

input
4 0 0 0 0 2 2 3 1 2 3 2 4 1 4 3 1 2 1 6 1 0 2 0 3 -1 4 -2 1 1 1 5
output
1 1 2 1 3 2

input
4 4 -5 -3 -1 2 2 3 1 2 3 2 4 1 4 3 1 2 1 6 1 0 2 0 3 -1 4 -2 1 1 1 5
output
1 1 1 3 1 1

Note

The first example can be shown like the following image:

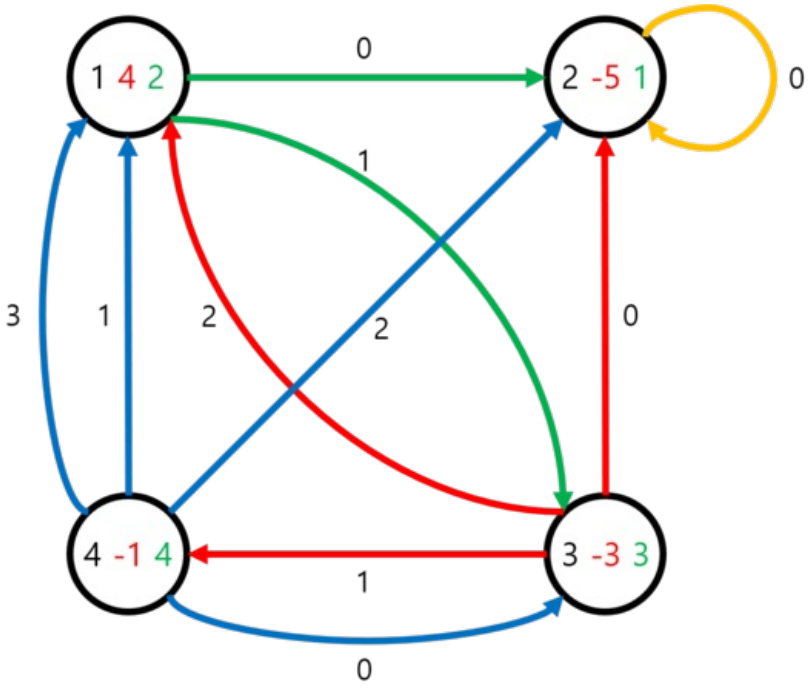


Three integers are marked on i-th vertex: i, k_i, and m_i respectively. The outgoing edges are labeled with an integer representing the edge number of i-th vertex.

The travel for each query works as follows. It is described as a sequence of phrases, each in the format "vertex (c after k_i added)".

- $1(0) \to 2(0) \to 2(0) \to \ldots$
- $2(0) \to 2(0) \to \ldots$
- $3(-1) \to 1(-1) \to 3(-1) \to \ldots$
- $4(-2) \to 2(-2) \to 2(-2) \to \ldots$
- $1(1) \to 3(1) \to 4(1) \to 1(1) \to \ldots$
- $1(5) \to 3(5) \to 1(5) \to \ldots$

The second example is same as the first example, except that the vertices have non-zero values. Therefore the answers to the queries also differ from the first example.



The queries for the second example works as follows:

- $1(4) \to 2(-1) \to 2(-6) \to \ldots$
- $2(-5) \to 2(-10) \to \ldots$
- $3(-4) \to 1(0) \to 2(-5) \to 2(-10) \to \ldots$
- $4(-3) \to 1(1) \to 3(-2) \to 4(-3) \to \ldots$
- $1(5) \to 3(2) \to 1(6) \to 2(1) \to 2(-4) \to \ldots$
- $1(9) \to 3(6) \to 2(1) \to 2(-4) \to \ldots$