15-295 Fall 2021 #14 Selection Round 2

A. Visiting Your Friends

2 s., 512 MB

There are two graphs defined over the same set of n nodes. There's a friendship graph, and a transportation graph. Both of these are undirected, unweighted, simple graphs. Initially the friendship graph has m edges and the transportation graph has k edges.

The set of nodes never changes, but new edges can be added over time to either graph. Edges are never removed. As the graphs evolve, there are queries that must be answered. A query specifies a vertex v, and it requires you to count the number of neighbors of v in the current friendship graph that are reachable from v in the current transportation graph.

Input

The first line contains three integers n, m, k — number of nodes, the initial number of edges in the friendship graph, and the initial number of edges in the transportation graph ($1 \le n \le 10^5, 0 \le m, k \le 10^5$).

The following $m + k$ lines contain two integers a and b each ($1 \le a, b \le n, a \ne b$) — an edge of a graph. The first m of them define the friendship graph, and the next k define the transportation graph.

The next line contains the integer q ($0 \le q \le 10^5$) — the number of requests and updates to be done. Each of the following q lines is a request or update.

- "`F` a b" adds an between a and b ($1 \le a, b \le n, a \ne b$) in the friendship graph. It is guaranteed that there was no edge (a, b) in the friendship graph before the update.
- "`T` a b" adds an between a and b ($1 \le a, b \le n, a \ne b$) in the transportation graph. It is guaranteed that there was no edge (a, b) in the transportation graph before the update.
- "`?` v" means that your program should print the number of neighbors of v ($1 \le v \le n$) in the current friendship graph that are reachable from v in the current transportation graph.

Output

For each request "`?` v" print the answer on a new line.

input
4 2 2 1 2 1 3 1 2 1 4 5 ? 1 F 4 1 ? 1 T 4 3 ? 1
output
1 2 3

B. A Two-Disk Cover

1 s., 256 MB

You're given a list L of n points in the plane. You're also told the location of the centers of two disks, c_1 and c_2. The goal is to determine the radii of the two disks r_1 and r_2 such that each of the points of L lies inside one or both of the disks. Compute the minimum possible value of $r_1^2 + r_2^2$ under these constraints.

Input

The first line contains four integers (x_1, y_1), (x_2, y_2), the location of the centers of the two disks. The second line contains an integer n ($1 \le n \le 10^5$), which is the number of points in the list L. Each of the next n lines contains two integers x and y, where (x, y) is a point in L. The coordinates af the centers of the disks and the points of L are all at most 1000 in absolute value.

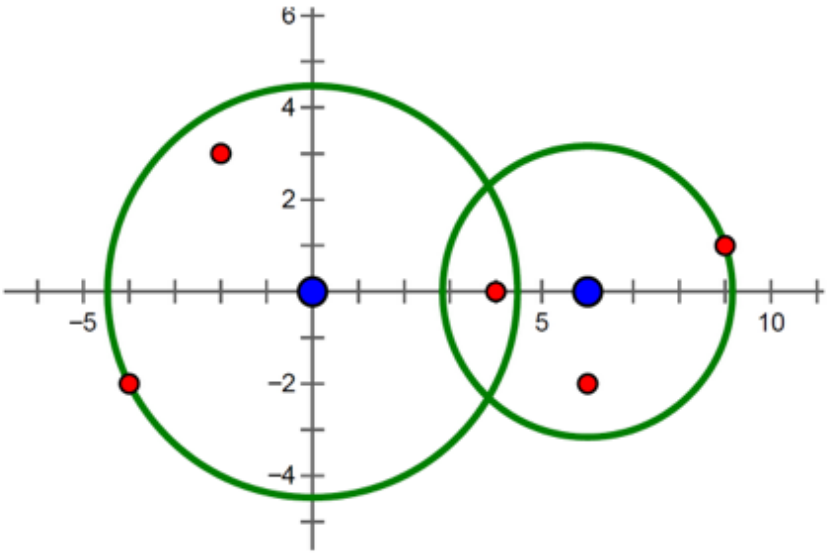
Output

Output one integer, the minimum possible value of $r_1^2 + r_2^2$, under the circumstances.

input
0 0 10 0 2 -3 3 10 0
output
18

input
0 0 6 0 5 -4 -2 -2 3 4 0 6 -2 9 1
output
30

The following figure illustrates sample 2. In the optimal solution the two radii are $\sqrt{20}$ and $\sqrt{10}$.



C. Particle Partition

1 s., 256 MB

There is an ordered list of n particles, the ith of which has a mass w_i and a maximum speed of s_i. The goal is to partition the list into groups of consecutive particles in such a way as to minimize the time it takes them to pass the obstacle.

To pass the obstacle the group must have a total weight of at most W. The length of the obstacle is L. The time it takes a group to pass an obstacle is L divide by the speed of the slowest particle in the group.

The first group enters the obstacle at time 0. After that group completes its pass through the obstacle the second group enters the obstacle. When it's finished, the third group enters, etc.

The completion time is defined as the time that the last group exits the obstacle.

Input

The first line contains three integers, $W \leq 2^{31} - 1$, the upper bound on the weight of a group, $L \leq 2^{31} - 1$, the length of the obstacle (in meters), and $n \leq 1000$, the number particles in the list.

Each of the next n lines contains two integers: $w_i \leq W$, the weight of each particle, and $1 \leq s_i \leq 2^{31} - 1$, the speed of each particle (in meters/second).

Output

The output is a single real number indicating the total time (in seconds), with at least eight digits after the decimal point.

Actually, the output should be in units of 1/60 seconds. See sample.

input
100 5 10 40 25 50 20 50 20 70 10 12 50 9 70 49 30 38 25 27 50 19 70

output
75.00000000

D. Equivalent Spanning Trees

3 s., 1024 MB

You're given a list of d weighted spanning trees on n nodes numbered $1, 2, \ldots, n$. A spanning tree T in this list defines a *bottleneck cost* $\mathrm{BC}_T(i, j)$ between each pair of nodes i and j – it's the minimum cost edge on the path from i to j in T.

Two spanning trees S and T are *equivalent* if for all $1 \le i < j \le n$ we have $\mathrm{BC}_S(i, j) = \mathrm{BC}_T(i, j)$.

The goal of this problem is to determine the equivalences class of each of the d spanning trees in the list.

Input

The first line contains two integers d and n with $1 \le d$, $2 \le n$, and $n \cdot d \le 10^6/2$.

Following this are $d \cdot (n - 1)$ lines. The first $n - 1$ of them define the first spanning tree, and so on. Each line is of the form $a\ b\ c$, which defines an edge from node a to node b of cost c. So $1 \le a, b \le n$, $a \ne b$, and $1 \le c \le 10^9$.

Output

The output is just one line with d integers. The ith number should be the minimum index j, where the jth spanning tree in the input is equivalent to the ith network in the input.

input
3 3 1 2 1 1 3 1 1 2 1 2 3 1 1 2 1 2 3 2
output
1 1 3

input
3 4 1 2 2 2 3 1 3 4 2 1 3 2 2 4 2 2 3 1 1 2 2 1 3 1 3 4 2
output
1 2 1

E. Simulate the Machine

4 s., 1024 MB

The machine has n registers $x_1, \ldots, x_n$, each containing an 8-bit byte. A program is a sequence of operations. Each operation is a pair (a, b), meaning that the machine does the assignment $x_a \leftarrow x_a \text{ OR } x_b$. (OR is the usual bitwise or operation.)

So, a program P is specified, along with the initial values of all the registers, and an integer s. We're going to run the program for s steps. If the program comes to the end and there are more steps to go, it just starts over again. This process continues until s steps have been done.

The goal is to efficiently compute the final value of all the registers, even though the number of steps you might have to simulate is insane.

Input

The first line contains three integers n, L, and s, with $1 \leq n, L \leq 2^{18}$ and $1 \leq s \leq 10^{18}$. L is the length of the program.

The next L lines contain the program. Each of these lines contains a and b, the register numbers of the operation. ($1 \leq a, b \leq n$).

The last line contains n integers in the range 0 to 255, which specify the initial values of the registers.

Output

Print out the final register values (after s steps of simulation) in the same format as the last line of of the input.

input
5 4 5 1 2 2 3 2 4 4 4 8 0 5 3 10
output
15 7 5 3 10

F. Not Quite Scrabble

2 s., 256 MB

It all starts with n Scrabble trays with letters in them. The trays are numbered 1 through n. The letters in the ith tray are represented by a string t_i. You're going to (try) to build a goal word g in your own Scrabble tray by raiding letters in the other Scrabble trays.

But there are some rules and regulations. For each tray i there is a limit l_i on the number of letters that can be stolen from it. Also, the letters have a cost associated with them. The letters in tray i cost i dollars to use.

Your goal is to compute the minimum total cost needed to build g.

Input

The input begins with your goal string g on a line by itself. The next line contains n ($1 \leq n \leq 100$) the number of trays you have to work from. Each of the next n lines contain a string t_i, followed by l_i.

The strings are all non-empty, and consist of lower case Latin letters. The total length of all the strings does not exceed 100.

Output

Print the minimum cost of a solution, or `-1` if there is no solution.

input
jjaze 3 jzj 2 aej 3 ja 10
output
8

input
wbwcwbw 4 wbw 2 bcc 1 cww 2 bbb 5
output
18

G. Simple Linkages

1 s., 256 MB

You're given a linkage system consisting of n links. The system is restricted to operate in two dimensions. One end of link 1 is pinned in a flexible manner to the point $(0, 0)$. The other end of link 1 is pinned to link 2, also in a flexible manner. The construction continues up to the nth link. Although the links are connected together, they do not otherwise interfere with each other.

The goal is to get the far end of link n as close as possible to some specified target point (x, y).

Input

The first line contains n, the number of links in the linkage. Each of the following n lines contain an integer L_i, the length of the ith link. The final link contains two more integers, the target point (x, y).

The bounds on these inputs are: $1 \leq n \leq 20$, $1 \leq L_i \leq 1000$, and the absolute values of the target coordinates do not exceed 2×10^4.

Output

The output consists of n lines, each containing two real numbers x_i and y_i, indicating the coordinates of the tip of the ith link. The length of each link, as derived from your numbers should not differ from the actual length by more than $.01$. Also, your solution's distance to the target should not differ by more than $.01$ from that of the correct solution.

input
3 5 3 4 5 3
output
4.114 -2.842 6.297 -0.784 5.000 3.000

input
2 4 2 -8 -3
output
-3.745 -1.404 -5.618 -2.107

H. Roller Coaster

1 second, 256 megabytes

Yihan recently received the job to design and test roller coasters for amusement parks. The safety and security of roller coasters are of great importance, so each design needs careful simulation to make sure everything is alright.

Hence, before a roller coaster is installed, it has to be run on a simulated track. A simulated track consists of n rails attached end-to-end with the beginning of the first rail fixed at elevation 0. Yihan can reconfigure the track at will by adjusting the elevation change over a number of consecutive rails. The elevation change over other rails is not affected. Each time rails are adjusted, the following track is raised or lowered as necessary to connect the track while maintaining the start at elevation 0. The figure on the next page illustrates the track reconfigurations for the sample input.

Each ride is initiated by launching the car with sufficient energy to reach height h. That is, the car will continue to travel as long as the elevation of the track does not exceed h, and as long as the end of the track is not reached. Given the record for all the day's rides and track configuration changes, compute for each ride the number of rails traversed by the car before it stops.

Internally, the simulator represents the track as a sequence of n elevation changes, one for each rail. The i-th number d_i represents the elevation change (in centimeters) over the i-th rail. Suppose that after traversing $i-1$ rails, the car has reached an elevation of h centimetres. After traversing i rails, the car will have reached an elevation of $h+d_i$ centimeters. Initially the rails are horizontal; that is, $d_i = 0$ for all i. Rides and reconfigurations are interleaved throughout the day. Each reconfiguration is specified by three numbers: a, b and D. The segment to be adjusted consists of rails a through b (inclusive). The elevation change over each rail in the segment is set to D. That is, $d_i = D$ for all $a \le i \le b$. Each ride is specified by one number h—the maximum height that the car can reach.

Your task is to write a program that reads from the standard input a sequence of interleaved reconfigurations and rides, and computes the number of rails traversed by the car for each ride.

Input

The first line of input contains one positive integer n — the number of rails, $1 \le n \le 10^9$. The following lines contain reconfigurations interleaved with rides, followed by an end marker. Each line contains one of:

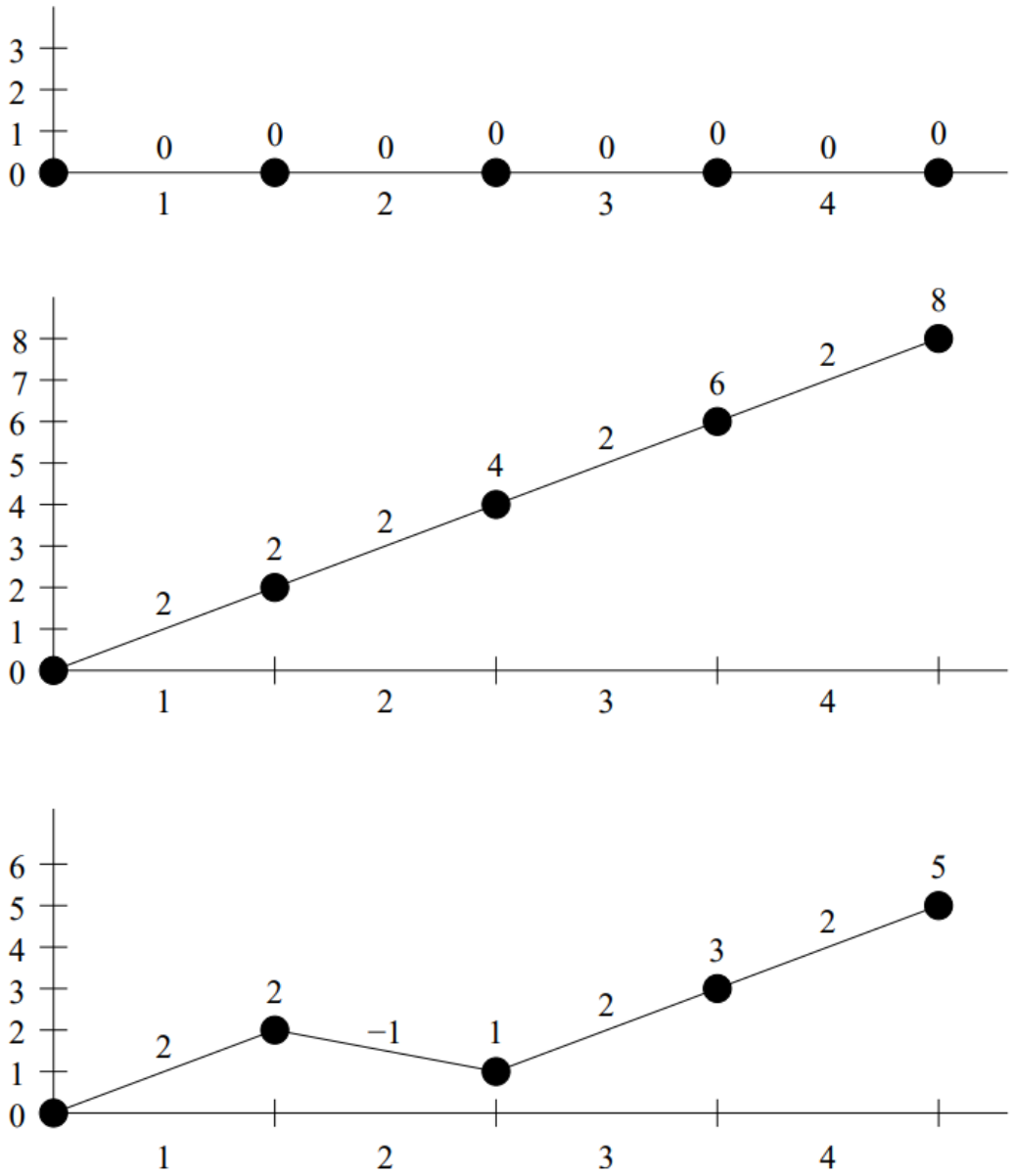
- Reconfiguration — a single letter "I", and integers a, b and D, all separated by single spaces ($1 \le a \le b \le n$, $-10^9 \le D \le 10^9$),
- Ride — a single letter "Q", and an integer h ($0 \le h \le 10^9$) separated by a single space,
- A single letter "E"—the end marker, indicating the end of the input data.

You may assume that at any moment the elevation of any point in the track is in the interval $[0, 10^9]$ centimeters. The input contains no more than 10^5 lines.

Output

The i-th line of output should consist of one integer—the number of rails traversed by the car during the i-th ride.

input
4 Q 1 I 1 4 2 Q 3 Q 1 I 2 2 -1 Q 3 E
output
4 1 0 3



Views of the track before and after each reconfiguration. The x axis denotes the rail number. The y axis and the numbers over points denote elevation. The numbers over segments denote elevation changes.