Problem A. Mines

Input file:	*standard input*
Output file:	*standard output*
Time limit:	10 seconds
Memory limit:	256 mebibytes

There are N mines on the number line. Mine i is at position p_i and has an explosion radius r_i. It initially costs c_i to detonate. If mine i is detonated, an explosion occurs on interval $[p_i - r_i, p_i + r_i]$ and all mines in that interval (inclusive of the endpoints) are detonated for free, setting off a chain reaction. You need to process Q operations of the form (m, c): Change the cost of detonating mine m to c. Output the minimum cost required to detonate all mines after each change. Note that each change is permanent.

Input

The first line contains integers N and Q ($1 \le N, Q \le 200\,000$). The next N lines contain information on the mines. The i-th of these lines contains integers p_i, r_i and c_i ($1 \le p_i, r_i, c_i \le 10^9$). The next Q lines each contains space separated integers m and c ($1 \le m \le N$, $1 \le c \le 10^9$).

Output

Output Q lines. The i-th line should contain the minimum cost required to detonate all mines after the i-th operation.

Example

standard input	standard output
4 2 1 1 1 6 3 10 8 2 5 10 2 3 1 1 4 11	4 6

Problem B. Balls

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

There are N balls on the number line, each of diameter 1, numbered from 1 through N. Ball i's leftmost point is located at position p_i. Additionally, there is an immovable wall located at position P. You have to process Q queries of one of the following forms:

- "1 x": Insert a new ball with its leftmost point at x. **If this spot is already occupied, do nothing**.
- "2": Roll the leftmost ball to the right. When a rolling ball (possibly after moving distance zero) collides with a stationary ball, it stops, and the stationary ball begins rolling in the same direction. Specifically, a rolling ball stops at the position 1 less than the position of the object it collided with. A ball stops when it reaches the wall.

Calculate the final positions of the balls.

Input

The first line contains three integers N, Q, and P: the initial number of balls, the number of queries, and the position of the wall ($1 \leq N, Q \leq 10^5$, $N \leq P \leq 10^9$).

The second line contains N integers $p_1, p_2, \ldots, p_N$ ($0 \leq p_i < P$). It is guaranteed the positions are distinct.

The next Q lines describe the queries and may have one of the following forms:

- "1 x" ($0 \leq x < P$)
- "2"

Output

Print out the final positions of the balls in increasing order on a single line, separated by spaces.

Examples

standard input	standard output
2 1 5 1 3 2	2 4
5 3 10 1 8 3 7 2 2 1 4 2	1 3 5 6 8 9

Problem C. Flip a Coin

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Alice and Bob have each picked a string containing only heads and tails. Then a fair coin is flipped until a sequence of consecutive flips matches one or both of the strings. Alice wins if her string appears first, and Bob wins if his appears first. It's possible that both of their strings appear at the same time. In that case the game is a tie.

Given the two strings, what is the probability of these three outcomes?

Input

The first line of input is Alice's string, and the second line is Bob's. These strings contain only `H`s and `T`s, and their lengths are between 1 and 20, inclusive.

Output

The output consists of three lines, each of which contains a single real number. They should be the probability that Alice wins, the probability that Bob wins, and the probability of a tie.

An answer will be accepted if it differs from the correct answer by at most 10^{-8}.

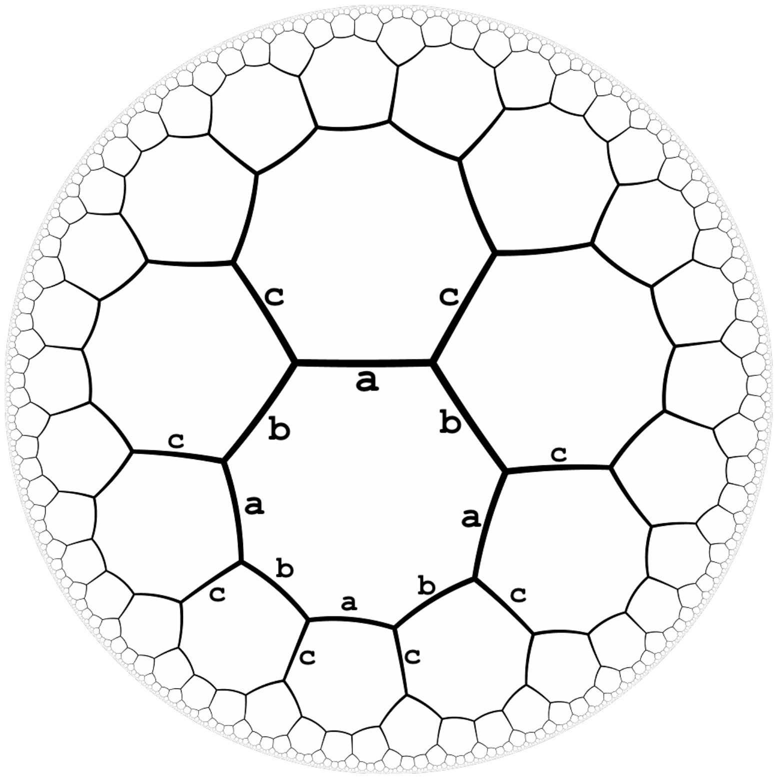
Examples

standard input	standard output
H T	0.500000000000 0.500000000000 0.000000000000
HHT HTH	0.666666666667 0.333333333333 0.000000000000
THH HH	0.000000000000 0.250000000000 0.750000000000

Problem D. Octagons

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Below is a picture of an infinite hyperbolic tessellation of octagons. If we think of this as a graph of vertices (of degree three), then there exists an isomorphism of the graph which maps any vertex x onto any other vertex y. Every edge is given a label from the set $\{\mathtt{a}, \mathtt{b}, \mathtt{c}\}$ in such a way that every vertex has all three types of edges incident on it, and the labels alternate around each octagon. Part of this labeling is illustrated in the diagram below.



So a path in this graph (starting from any vertex) can be specified by a sequence of edge labels. Your job is to write a program which, given a sequence of labels such as "`abcbcbcabcaccadb`", returns "`closed`" if the path ends on the same vertex where it starts, and returns "`open`" otherwise.

Input

The input is a string of length at least 1 and at most 100 000 consisting of letters "`a`", "`b`" and "`c`".

Output

The output should be one line with one word: either "`closed`" or "`open`".

Examples

standard input	standard output
abababab	closed
abcbcbcbcba	open

Problem E. Tree Paths

Input file:	*standard input*
Output file:	*standard output*
Time limit:	15 seconds
Memory limit:	512 mebibytes

There is a tree of N vertices numbered 1 to N. A path is a sequence of distinct vertices $(v_1, \ldots, v_k)$ such that $k \geq 1$, $v_i v_{i+1}$ is an edge for all $1 \leq i \leq k-1$, and $v_1 \leq v_k$.

Count the number of paths such that the vertices $v_1, \ldots, v_k$ form a contiguous range, or more formally, the set $\{v_1, \ldots, v_k\} = \{a, a+1, \ldots, b\}$ for some integers $a \leq b$.

Input

The first line contains an integer N ($1 \leq N \leq 50\,000$). The next $N-1$ lines contain the edges of the tree. The i-th of these lines contains two space-separated integers u_i and v_i ($1 \leq u_i, v_i \leq N$) denoting that $u_i v_i$ is an edge. It is guaranteed that the given graph is a tree.

Output

On a single line output the desired number of paths.

Example

standard input	standard output
3 1 2 1 3	5

Note

The paths are (1), (2), (3), $(1, 2)$, and $(2, 1, 3)$.

Problem F. Very New York

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

The year is 2211. Very New York is a city on Mars. The streets of the city make up a perfect grid. Each intersection in the city can be specified using a pair (x, y) of integers. The distance between two intersections (x_1, y_1) and (x_2, y_2) equals $|x_1 - x_2| + |y_1 - y_2|$.

An investor is interested in building a hotel in the city. Since hotel owners love to advertise hotels using phrases of the form "150 restaurants within half a mile", the investor wants to learn the number of restaurants within a specific distance from each of the prospective locations.

Input

The first line contains R, the number of restaurants in the city ($0 \leq R \leq 100\,000$). The next R lines describe the coordinates of one restaurant each. Each of these lines contains two integers x and y ($1 \leq x, y \leq 1\,000\,000$).

The $(R+2)$-nd line contains one integer Q which is the number of queries ($1 \leq Q \leq 100\,000$). The next Q lines contain one query each. Each query consists of a triple of integers x, y, and d ($1 \leq x, y, d \leq 1\,000\,000$).

Output

The output should consist of Q lines, each containing a single integer. The i-th line should contain the answer to the i-th query (x, y, d): the number of restaurants at distance at most d from (x, y).

Examples

<table>
<tr><th>standard input</th><th>standard output</th></tr>
<tr><td><pre>3
2 2
3 1
4 1
2
3 2 1
4 2 1</pre></td><td><pre>2
1</pre></td></tr>
<tr><td><pre>5
3 2
3 6
4 6
2 4
4 6
2
4 4 2
3 4 3</pre></td><td><pre>3
5</pre></td></tr>
</table>

Problem G. Sheep

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

Every day a shepherd guards a flock of sheep on an infinite one-dimensional pasture. The shepherd takes the flock to the pasture in the morning at time 0, and he takes the flock to the barn in the evening at time T. The locations of both the shepherd and each sheep on the pasture throughout the day can be described as functions from $[0, T]$ to $\mathbb{R}$.

After thousands of days of observation, the shepherd realized that every wolf that wants to attack his flock chooses a sufficiently lonely sheep. If $h : [0, T] \to \mathbb{R}$ describes the location of the shepherd throughout the day, then the loneliness of a specific sheep with the location described by $s : [0, T] \to \mathbb{R}$ is defined as

$$L(s, h) = \left(\max_{t \in [0,T]} (s(t) - h(t)) \right)^2 + \left(\min_{t \in [0,T]} (s(t) - h(t)) \right)^2 .$$

No wolf is infinitely brave. If the loneliness of a sheep is below some threshold specific for each wolf, the wolf does not attack the sheep.

The shepherd is wondering if it is possible to save all the sheep, and therefore, he wants to follow a trajectory that minimizes the loneliness of the most lonely sheep. Sheep are very simple animals so the location of each of them can always be described using a linear function $s(t) = a \cdot t + b$, for some a and b. While as the above equation shows, the shepherd is a very smart guy, he is also slightly lazy, so he would like to follow a linear function as well.

Given locations of all sheep, and assuming that the shepherd trajectory is a linear function as well, compute the lowest possible loneliness of the most lonely sheep.

Input

The first line contains one positive integer T ($1 \le T \le 100$). This is the time when the shepherd takes the flock back to the barn. The second line contains one positive integer n ($1 \le n \le 100\,000$). This is the number of sheep. Each of the next n lines contains two integers a and b, describing the trajectory $s(t) = a \cdot t + b$ of one sheep. It is guaranteed that $|a| \le 100\,000$ and $|b| \le 100\,000$. No two sheep have the same trajectory.

Output

Output one line containing one real number: the minimum loneliness of the most lonely sheep. You answer can differ from the correct one by at most 0.01. In all tests we prepared, the answer is at most 10^{10}.

Examples

standard input	standard output
5 2 10 5 10 14	40.5000
3 4 3 0 3 4 4 -1 6 -8	38.2500

Problem H. Bin Packing

Input file:	*standard input*
Output file:	*standard output*
Time limit:	12 seconds
Memory limit:	256 megabytes

You're given a collection of n objects of weights $w_1, w_2, \ldots, w_n$. You have to pack all n objects into the minimum number of bins under the constraint that the total weight of all the objects in any bin is bounded by S.

Input

The first line contains a pair of integers n and S, where $1 \leq n \leq 24$ and $1 \leq S \leq 10^8$. The second line contains $w_1, w_2, \ldots, w_n$, where $1 \leq w_i \leq S$.

Output

The output is just the minimum number of bins required to pack the given objects.

Example

standard input	standard output
4 10 5 6 3 7	3

Note

The objects can be packed into three bins of size 10 as follows: [5,3], [6], [7]. It is impossible to pack them into two bins because their total size is 21.

Problem I. Statistics

Input file:	*standard input*
Output file:	*standard output*
Time limit:	6 seconds
Memory limit:	256 mebibytes

It is time to select courses for your last semester of university. There are N courses available, numbered from 1 through N. Course i is worth v_i units, and you need at least V units to graduate. Because you want more time to train for ACM ICPC, you want to choose a subset of courses that gives exactly V units. Also, you want to choose the least number of courses possible to minimize the amount of time spent walking around campus. You call such a subset a **good schedule**.

You are having trouble choosing among all possible good schedules, so you turn to statistics for help. For each good schedule, viewing it as a list of the units of its courses, you calculate:

- a, the average value.
- b, the median value. (In case the list's length is even, use the smaller of the two middle values.)
- c, the maximum number of times a single value appears.
- d, the difference between the maximum value and the minimum value.

For each of a, b, c, and d, find its minimum over all good schedules.

Input

The first line contains two integers, N and V: the number of courses and the number of units you need to graduate ($1 \le N, V \le 5000$).

The next line contains N integers $v_1, v_2, \ldots, v_N$, the number of units in each of the available courses ($1 \le v_i \le V$).

Output

Print out four space-separated real numbers: the minimum a, b, c, and d over all good schedules, in that order. If there are no good schedules, print a single integer -1 instead. Your answer must have an absolute or relative error less than 10^{-6}.

Examples

standard input	standard output
6 15 6 1 13 5 4 1	5.000000000 1 1 2
3 7 3 1 2	-1

Problem J. Zigzag

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

We say that a sequence of numbers $x(1), x(2), \ldots, x(k)$ is *zigzag* if no three of its consecutive elements form a nonincreasing or nondecreasing sequence. More precisely, for all $i = 1, 2, \ldots, k-2$, either $x(i+1) < x(i)$ and $x(i+1) < x(i+2)$ or $x(i+1) > x(i)$ and $x(i+1) > x(i+2)$.

You are given two sequences of numbers $a(1), a(2), \ldots, a(n)$ and $b(1), b(2), \ldots, b(m)$. The problem is to compute the length of the longest common zigzag subsequence. In other words, you are going to remove elements from the two sequences so that they are equal, and what remains from each sequence is a zigzag sequence. If the minimum number of elements you have to remove is k, then your answer is $m + n - k$.

Input

The first line contains the length of the first sequence n ($1 \le n \le 2000$) followed by n integers $a(1), a(2), \ldots, a(n)$. The second line contains the length of the second sequence m ($1 \le m \le 2000$) followed by m integers $b(1), b(2), \ldots, b(m)$. All $a(i)$ and $b(i)$ are from the range $[-10^9, 10^9]$.

Output

Output one line containing the length of the longest common zigzag subsequence of a and b.

Example

standard input	standard output
5 1 2 5 4 3 5 4 1 2 5 3	3

Problem K. Knapsack

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

```
 To fit the best stuff in a sack
with limits on what you can pack
      recursion will do
       if memoized too
 but DP puts less on the stack.
```

You are given a classic knapsack problem: a set of elements $(w_1, v_1), \ldots, (w_n, v_n)$ and a capacity W. Solve

$$\max_{\substack{S \subseteq [n] \\ \sum_{i \in S} w_i \leq W}} \sum_{i \in S} v_i.$$

Here, $[n]$ is the list of integers from 1 through n.

You are guaranteed that $0 \leq n \leq 500$, $0 \leq w_i \leq W \leq 10^{17}$, and $0 \leq v_i \leq 10^{16}$. Furthermore, you are guaranteed that the w_i have been "smoothed" as described in the next paragraph.

A problem instance complying with the above bounds is constructed. Then a randomizing filter is applied to the problem instance as follows: Let B be the weight of the element of maximum weight. Now the following update is applied to each weight:

$$w_i \leftarrow \min(w_i + \text{rand}(\lfloor .05B \rfloor), W)$$

Here $\text{rand}(A)$ is a function that generates a uniform random integer in $[0, A-1]$. This process is applied only to the w_i, not to W or any of the other values.

Input

The first line contains the two space-separated integers n and W. Each of the next n lines contains two space-separated integers w_i and v_i. The element weights were generated as described above. First the items are chosen in compliance with all the bounds. Then the smoothing algorithm is applied. The result is what your program will receive as input.

Output

Output one line with the value of the given knapsack problem instance.

Examples

standard input	standard output
3 5 1 2 2 3 3 4	7
3 302000 100588 2 200421 1000 30036 1	1002