Problem A. Fading Wind

Source file name:	Fading.c, Fading.cpp, Fading.java, Fading.py
Input:	`Standard`
Output:	`Standard`

You're competing in an outdoor paper airplane flying contest, and you want to predict how far your paper airplane will fly. Your design has a fixed factor k, such that if the airplane's velocity is at least k, it will rise. If its velocity is less than k it will descend.

Here is how your paper airplane will fly:

- You start by throwing your paper airplane with a horizontal velocity of v at a height of h. There is an external wind blowing with a strength of s.
- While $h > 0$, repeat the following sequence:
  - Increase v by s. Then, decrease v by $\max\left(1, \left\lfloor \frac{v}{10} \right\rfloor\right)$. Note that $\left\lfloor \frac{v}{10} \right\rfloor$ is the value of $\frac{v}{10}$, rounded down to the nearest integer if it is not an integer.
  - If $v \geq k$, increase h by one.
  - If $0 < v < k$, decrease h by one. If h is zero after the decrease, set v to zero.
  - If $v \leq 0$, set h to zero and v to zero.
  - Your airplane now travels horizontally by v units.
  - If $s > 0$, decrease it by 1.

Compute how far the paper airplane travels horizontally.

Input

The single line of input contains four integers h, k, v, and s ($1 \leq h,\ k,\ v,\ s \leq 10^3$), where h is your starting height, k is your fixed factor, v is your starting velocity, and s is the strength of the wind.

Output

Output a single integer, which is the distance your airplane travels horizontally. It can be shown that this distance is always an integer.

Example

Input	Output
1 1 1 1	1
2 2 2 2	9
1 2 3 4	68
314 159 265 358	581062

Problem B. Creative Accounting

Source file name:	Creative.c, Creative.cpp, Creative.java, Creative.py
Input:	`Standard`
Output:	`Standard`

When accounting for the profit of a business, we can divide consecutive days into fixed-sized segments and calculate each segment's profit as the sum of all its daily profits. For example, we could choose seven-day segments to do our accounting in terms of weekly profit. We also have the flexibility of choosing a segment's starting day. For example, for weekly profit we can start a week on a Sunday, Monday, or even Wednesday. Choosing different segment starting days may sometimes change how the profit looks on the books, making it more (or less) attractive to investors.

As an example, we can divide ten consecutive days of profit (or loss, which we denote as negative profit) into three-day segments as such:

$$3, 2, -7|5, 4, 1|3, 0, -3|5$$

This gives us four segments with profit -2, 10, 0, 5. For the purpose of this division, partial segments with fewer than the fixed segment size are allowed at the beginning and at the end. We say a segment is profitable if it has a strictly positive profit. In the above example, only two out of the four segments are profitable.

If we try a different starting day, we can obtain:

$$3, 2| - 7, 5, 4|1, 3, 0| - 3, 5$$

This gives us four segments with profit 5, 2, 4, 2. All four segments are profitable, which makes our business look much more consistent.

You're given a list of consecutive days of profit, as well as an integer range. If we can choose any segment size within that range and any starting day for our accounting, what is the minimum and maximum number of profitable segments that we can have?

Input

The first line of input has three space-separated integers n, l and h ($1 \leq l \leq h \leq n \leq 3 \cdot 10^4$, $h - l \leq 10^3$), where n is the number of days in the books, l is the minimum possible choice of segment size, and h is the maximum possible choice of segment size.

Each of the next n lines contains a single integer p ($-10^4 \leq p \leq 10^4$). These are the daily profits, in order.

Output

Output on a single line two space-separated integers min and max, where min is the minimum number of profitable segments possible, and max is the maximum number of profitable segments possible. Both min and max are taken over all possible choices of segment size between l and h and all possible choices of starting day.

Example

Input	Output
10 3 5 3 2 -7 5 4 1 3 0 -3 5	2 4

Problem C. Chocolate Chip Fabrication

Source file name:	Chocolate.c, Chocolate.cpp, Chocolate.java, Chocolate.py
Input:	`Standard`
Output:	`Standard`

You are making a chocolate chip cookie using a machine that has a rectangular pan composed of unit squares. You have determined the shape of your cookie, which occupies some squares in that area. Each square of your cookie must be chocolate chipified.

To make the cookie you will repeatedly perform the following two steps:

1. You place cookie dough in some unit squares.
2. You expose the cookie dough to a shallow chocolate chip solution. Any cookie dough square that does not have all four adjacent squares (up, down, left, right) filled with cookie dough becomes chocolate chipified. Note that any cookie dough in a square on the boundary of the pan always gets chipified.

The following example shows how to make a cookie of the shape shown on the left (s):

```
(s)     (a1)    (a2)    (b1)    (b2)
-X-X-   -D-D-   -C-C-   -C-C-   -C-C-
XXXXX   -D-D-   -C-C-   DCDCD   CCCCC
XXXXX   -DDD-   -CCC-   DCCCD   CCCCC
-XXX-   --D--   --C--   -DCD-   -CCC-
--X--   -----   -----   --D--   --C--
```

First you place cookie dough in 8 squares (a1). All squares become chipified after the first solution exposure (a2). You place cookie dough in 8 more squares (b1). The second exposure makes every square chipified and completes the cookie (b2).

Your chocolate chip solution is expensive, so you want to ensure that you perform the exposure as few times as possible. Given a cookie shape, determine the minimum number of chocolate chip solution exposures required to make the cookie.

Input

The first line of input contains two integers n and m ($1 \leq n,\ m \leq 10^3$), indicating the pan has n rows and m columns of unit squares.

Each of the next n lines contains a string of exactly m characters, where each character is either "`X`", representing a square occupied by your cookie, or "`-`", representing an empty square.

The shape of your cookie occupies at least one square. Note that the shape may consist of multiple pieces that are disconnected.

Output

Output the minimum number of chocolate chip solution exposures required to make your cookie.

Example

Input	Output
5 5 -X-X- XXXXX XXXXX -XXX- --X--	2
4 5 --XXX --X-X X-XXX XX---	1
5 5 XXXXX XXXXX XXXXX XXXXX XXXXX	3

Problem D. Everything Is A Nail

Source file name:	Everything.c, Everything.cpp, Everything.java, Everything.py
Input:	`Standard`
Output:	`Standard`

As an employee of the Iffy Colossal Pinnacle Construction (ICPC) company building a very tall skyscraper, you have a number of tasks to complete high above the ground in a specific order. You can always choose to skip a task, but you fear that doing so too many times might cause some catastrophic failure of the building. You cannot revisit or complete a task once it has been skipped.

Each task is a nail, a screw, or a bolt. You have three tools: a hammer (works on nails), a screwdriver (works on screws), and a wrench (works on bolts). When you start a new task you can choose to switch your tool out by dropping it (hopefully no one was below you at the time), but when you do so you permanently lose the dropped tool.

Given the list of tasks in the order they should be completed, determine the maximum number of tasks that can be completed. You may choose to use any tool as the initial tool.

Input

The first line of input contains an integer n ($1 \leq n \leq 3 \cdot 10^5$), which is the number of tasks you need to complete.

Each of the next n lines contains a single integer t ($0 \leq t \leq 2$). These are the tasks, in order. Each task is one of 0 (nail), 1 (screw), or 2 (bolt).

Output

Output a single integer, which is the maximum number of tasks that can be completed.

Example

Input	Output
10 1 1 1 0 0 0 0 2 2 2	10
10 0 1 2 0 1 2 0 1 2 0	5

Problem E. Exponent Exchange

Source file name:	Exponent.c, Exponent.cpp, Exponent.java, Exponent.py
Input:	`Standard`
Output:	`Standard`

Alice and Bob are playing a cooperative game. They hold b^p dollars between them, for given integers b and p. Alice initially holds x dollars, and Bob holds $b^p - x$ dollars. Alice and Bob want to consolidate their money, so one person holds all the money.

In each transaction, one player can choose to send the other player b^y dollars, for some integer y with $0 \leq y < p$. But each player wants to initiate as few transactions as possible. They are willing to cooperate such that the player that initiates the most transactions (the busiest player), initiates as few as possible.

Alice and Bob want to know the fewest number of transactions that the busiest player needs to initiate to complete the transfer.

Input

The first line of input contains two integers b ($2 \leq b \leq 100$) and p ($2 \leq p \leq 1000$), where b is the base, and p is the number of digits.

The next line contains p integers x_{p-1}, x_{p-2}, ..., x_0, separated by spaces, with $0 \leq x_i < b$ and $0 < x_{p-1}$. These are the base-b digits of the value of x, with the most significant digit first. Specifically, $x = \sum_{0 \leq i < p} b^i x_i$. Note that they are given in order from the highest power to the lowest. For example, in the sample, 4 2 7 8 6 with $b = 10$ represents the base 10 number $42,786$.

Output

Output a single integer, which is the minimum number of transactions the busiest player must initiate to transfer all the money to either Alice or Bob.

Example

Input	Output
10 5 4 2 7 8 6	7

Problem F. Family Visits

Source file name:	Family.c, Family.cpp, Family.java, Family.py
Input:	`Standard`
Output:	`Standard`

You are a college student living on your own. However, your doting family still likes to visit you, and they often stop by to check on your room at night before going to dinner. Your family will be worried if they find a mess in your room. Therefore you make an effort to ensure that they never see a mess in your room when visiting at night. You have some free time each afternoon that allows you to clean up, but the amount of free time varies each day due to prior commitments.

Luckily, your schedule is planned out well. You know exactly how big of a mess you will make each morning, how much mess you can clean each afternoon, and on which nights your family will stop by. Since you are lazy, you want to spend as few afternoons as possible cleaning such that your family will always see a room without any mess. You may assume that your room starts completely clean, and any mess that is not cleaned remains until it is cleaned.

Input

The first line of input contains two integers, n and d ($1 \leq d \leq n \leq 10^3$), where n is the number of days in your schedule and d is the number of days your family will visit.

Each of the next n lines contains two integers m and c ($0 \leq m,\ c \leq 10^3$). For each day, in order, m is the amount of mess you make in the morning, and c is the amount you can clean in the afternoon.

Each of the next d lines contains a single integer v ($1 \leq v \leq n$). These are the days on which your family will visit, and they are listed in strictly increasing order.

Output

Output the smallest number of afternoons you have to spend cleaning to ensure your family will never see a mess. If it is not possible, output -1.

Example

Input	Output
6 2 1 2 2 1 1 4 3 2 3 6 2 3 3 6	3
10 5 12 10 0 2 7 1 1 8 3 4 3 4 2 3 1 2 10 1 7 5 2 4 5 6 8	7

Problem G. Eroding Pillars

Source file name:	Eroding.c, Eroding.cpp, Eroding.java, Eroding.py
Input:	`Standard`
Output:	`Standard`

You've landed yourself in a real mess, or more accurately a cave filled with dilapidated and unstable pillars. You are luckily standing on a solid rock in the middle of the cave. You know there is a valuable artifact on one of the pillars in the cave, but you aren't sure which one yet. While you wait for the results of a scan of the cave, you start building a robot to help you retrieve the artifact.

The robot you build will be light, and it will assuredly be able to land and jump from any pillar at least once. To build the robot, you need to determine how powerful it needs to be in terms of its jump distance. If the robot is too weak, then it might not be able to reach a pillar. If the robot is too strong, then it will cause a lot of damage when jumping and landing.

Luckily you have already mapped out where all the pillars are located with respect to your starting point at $(0,\ 0)$. You don't know yet which pillar contains the artifact, and you need to finish building the robot first before the scan finishes.

Given the locations of the pillars, you'd like to determine the smallest jump distance to guarantee that your robot can reach any pillar and return back to the start, without landing on a pillar twice.

Input

The first line of input contains a single integer n ($1 \leq n \leq 10^3$), which is the number of pillars.

Each of the next n lines contains two integers, x and y ($-10^9 \leq x,\ y \leq 10^9$). These are the $(x,\ y)$ coordinates of the pillars. All pillar locations will be distinct, and no pillar will be at $(0,\ 0)$.

Output

Output a single number, which is the minimum jump distance needed to guarantee that a robot could reach any pillar and return back to your starting point, without landing on a pillar twice. Your answer will be accepted if it has an absolute error of at most 10^{-6}.

Example

Input	Output
2 1 1 1 0	1.414213562373095
8 1 1 0 1 1 0 2 0 0 2 2 1 1 2 2 2	1.0

Problem H. Triangle Containment

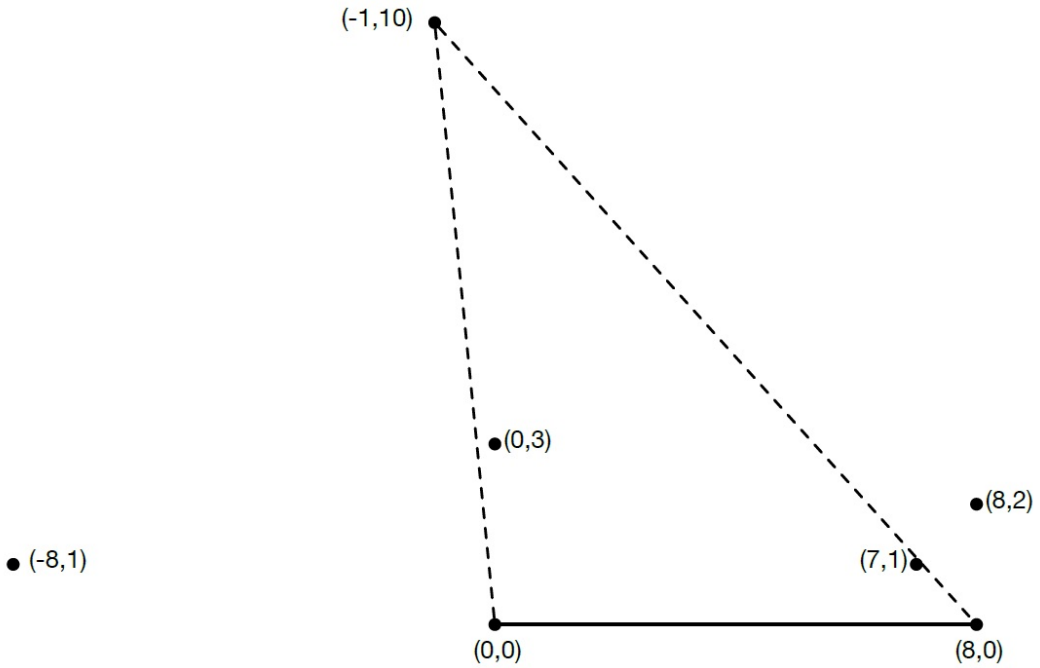
Source file name:	Triangle.c, Triangle.cpp, Triangle.java, Triangle.py
Input:	`Standard`
Output:	`Standard`

You recently discovered there is treasure buried on your farm land. A **lot** of treasure! You quickly decide to put a fence around the land.

Alas, you have but a single fence post! You will have to drive to town to get more fencing material. But you can't just leave the land as open as it is, so you decide to create a makeshift fence to protect some of the treasure while you are gone. You will place the post in the ground and run some wire in a straight line between two sides of your barn wall and the fence post to section off a triangular area. Also, the ground is very hard: only places that were dug up to bury a treasure are soft enough for you to quickly place the fence post.

To figure out the best option, you first calculate the following. For each of the treasures in your field, if you were to place the fence post at that treasure and complete the fence as described, then what is the total value of all treasures that would be enclosed by the fence? Note that the treasure under the post you place is not considered enclosed by the fence (it might not be safe since someone could dig around the post).

Example Input 1 is illustrated below. The triangle that includes the point $(-1,\ 10)$ encloses exactly two other treasure points which have total value $4 + 8 = 12$.



Input

The first line of input contains two integers n ($1 \le n \le 10^5$) and x ($1 \le x \le 10^9$), where n is the number of treasure points and x fixes the two corners of the barn wall at locations $(0,\ 0)$ and $(x,\ 0)$.

Each of the next n lines contains three integers x, y, and v ($-10^9 \le x \le 10^9$, $1 \le y \le 10^9$, and $1 \le v \le 10^9$) giving the location $(x,\ y)$ and value v of one of the treasure points. All of these points are distinct. It is also guaranteed that for each point, the triangle formed with the barn wall will not contain any other treasure point on its boundary.

Output

Output n lines, one for each treasure point in the order of the input. For each point output a single integer, which is the total value of all points in the interior of the triangle that point forms with the barn wall. Note that the value of the point itself should be excluded from this sum.

Example

Input	Output
5 8 -8 1 1 -1 10 2 0 3 4 7 1 8 8 2 16	0 12 0 0 8
6 6 0 1 1 2 3 10 2 5 100 3 1 1000 3 5 10000 4 5 100000	0 1000 1010 0 1010 1000

Problem I. Streets Ahead

Source file name:	Streets.c, Streets.cpp, Streets.java, Streets.py
Input:	`Standard`
Output:	`Standard`

International Connecting Passage Causeway is a long, rutted two-way country road crossed by streets at different points.

There are many drivers, and each will drive along the country road starting at some intersection and ending at some other intersection. For each driver, how many intersections will they drive through?

Input

The first line contains two integers, n ($2 \leq n \leq 10^5$) and q ($1 \leq q \leq 10^5$), where n is the number of cross streets and q is the number of drivers.

Each of the next n lines contains a string of at most ten lowercase letters representing the name of one of the streets that crosses the country road. All street names are unique. Driving along the country road in some direction, one sees these streets in exactly the order provided.

Each of the next q lines contains two strings of at most ten lowercase letters representing the start and end intersection for each driver. Queries will be between distinct streets.

Output

Output q lines, the i-th line containing the number of intersections that the i-th driver drives through.

Example

Input	Output
3 3 first second third first second third first second third	0 1 0

Problem J. Sun and Moon

Source file name:	Sun.c, Sun.cpp, Sun.java, Sun.py
Input:	`Standard`
Output:	`Standard`

You recently missed an eclipse and are waiting for the next one! To see any eclipse from your home, the sun and the moon must be in alignment at specific positions. You know how many years ago the sun was in the right position, and how many years it takes for it to get back to that position. You know the same for the moon. When will you see the next eclipse?

Input

The input consists of two lines.

The first line contains two integers, d_s and y_s ($0 \leq d_s < y_s \leq 50$), where d_s is how many years ago the sun was in the right position, and y_s is how many years it takes for the sun to be back in that position.

The second line contains two integers, d_m and y_m ($0 \leq d_m < y_m \leq 50$), where d_m is how many years ago the moon was in the right position, and y_m is how many years it takes for the moon to be back in that position.

Output

Output a single integer, the number of years until the next eclipse. The data will be set in such a way that there is not an eclipse happening right now and there will be an eclipse within the next $5,000$ years.

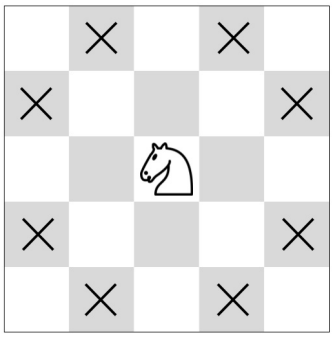
Example

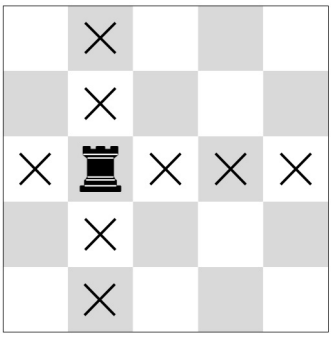
Input	Output
3 10 1 2	7

Problem K. Lone Knight

Source file name:	Knight.c, Knight.cpp, Knight.java, Knight.py
Input:	`Standard`
Output:	`Standard`

In the game of chess, a knight moves as shown in the picture below; each move is one square horizontally and two squares vertically or two squares horizontally and one square vertically. A rook can move any number of squares horizontally or vertically, but not both in the same move. If a square can be reached by a rook in one move, that square is said to be attacked by the rook.





Consider an infinite chess board, with squares that can be indexed by integer coordinates. There is a white knight on the board on a square, and it wants to go to another square. However, there are also a number of black rooks on the board. The knight can make as many moves as it needs to get to its target square, but it cannot stop on a square that is attacked by or occupied by a rook. The rooks don't move.

Can the white knight reach its target square? You are to answer that question many times!

Input

The first line of input contains two integers n and q ($1 \leq n,\ q \leq 10^3$), where n is the number of black rooks and q is the number of queries.

Each of the next n lines contains two integers x and y. This indicates that there is a black rook at $(x,\ y)$. No two rooks share the same square.

Each of the next q lines contains four integers x_s, y_s, x_t and y_t. This is a query, where the white knight starts at square $(x_s,\ y_s)$ and wants to move to square $(x_t,\ y_t)$.

All square coordinates in the input are no larger than 10^9 in absolute value. It is guaranteed that in every query the knight's initial and target squares are not attacked by or occupied by any rook, and the target square is not the same as the initial square.

Output

For each query, output on a single line 1 if the knight can reach the target square, or 0 otherwise.

Example

Input	Output
6 6 10 14 1 0 0 1 4 9 9 13 5 9 2 2 3 4 2 2 2 4 2 2 6 4 2 2 2 10 7 11 6 2 6 2 8 12	1 0 0 1 0 1
8 10 0 0 1 1 5 5 8 8 11 11 14 14 18 18 19 19 17 10 13 9 15 15 15 9 7 3 17 4 15 15 12 3 9 17 3 3 4 4 9 4 12 12 2 6 10 15 6 6 15 17 4 16 -1000000000 -999999999 15 7	1 0 0 0 0 0 0 0 0 0

Problem L. Branch Manager

Source file name:	Branch.c, Branch.cpp, Branch.java, Branch.py
Input:	`Standard`
Output:	`Standard`

You are managing a transportation network of one-way roads between cities. People travel through the transportation network one by one in order all starting from the same city, and each person waits for the person before them to stop moving before starting. The people follow a simple algorithm until they reach their destination: they will look at all the outgoing roads from the current city, and choose the one that leads to the city with the smallest label. A person will stop when they either reach their destination, or reach a city with no outgoing roads. If at any point someone fails to reach their destination, the rest of the people still waiting in line will leave.

Before each person enters the transportation network, you can permanently close down any subset of roads to guarantee they reach their destination. The roads that you choose to close down will not be available for future people.

There are n cities, labeled from 1 to n. There are $n-1$ directed roads, and each road will always be from a lower labeled city to a higher labeled one. The network will form a rooted tree with city 1 as the root. There are m people that want to travel through the network. Each person starts from city 1, and has a specific destination city d in mind. These people will line up in the given order. What is the maximum number of people you can route correctly to their destination if you close roads optimally?

Input

The first line of input contains two integers n and m ($2 \leq n,\ m \leq 2 \cdot 10^5$), where n is the number of cities and m is the number of people.

Each of the next $n-1$ lines contains two integers a and b ($1 \leq a < b \leq n$), denoting a directed road from city a to b. These roads will describe a rooted tree with city 1 as the root.

Each of the next m lines contains a single integer d ($2 \leq d \leq n$), denoting the destination city of the next person in line.

Output

Output a single integer, which is the maximum number of people you can route to the correct destination.

Example

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

Problem M. Alchemy

Source file name:	Alchemy.c, Alchemy.cpp, Alchemy.java, Alchemy.py
Input:	`Standard`
Output:	`Standard`

You just finished day one of your alchemy class! For your alchemy homework, you have been given a string of lowercase letters and wish to make it a palindrome. You're only a beginner at alchemy though, so your powers are limited. In a single operation, you may choose exactly two adjacent letters and change each of them into a different lowercase letter. The resulting characters may be the same as or different from one another, so long as they were both changed by the operation.

Formally, if the string before the operation is s and you chose to change characters s_i and s_{i+1} to produce string t, then $s_i \neq t_i$ and $s_{i+1} \neq t_{i+1}$ must be true, but $t_i = t_{i+1}$ is permitted.

Compute the minimum number of operations needed to make the string a palindrome.

Input

The single line of input contains a string of n ($2 \leq n \leq 100$) lowercase letters, the string you are converting into a palindrome.

Output

Output a single integer, which is the minimum number of operations needed to make the string a palindrome.

Example

Input	Output
ioi	0
noi	1
ctsc	1
fool	2
vetted	2