Problem A. Air Race

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are competing in a toy airplane flying contest, and you want to predict how far your 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 airplane will fly:

- You start by throwing your 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:

1. Increase v by s. Then, decrease v by $\max(1, [v/10])$. Note that $[v/10]$ is the value of $v/10$, rounded down to the nearest integer if it is not an integer.
2. If $v \geq k$, increase h by one.
3. If $0 < v < k$, decrease h by one. If h is zero after the decrease, set v to zero.
4. If $v \leq 0$, set h to zero and v to zero.
5. Your airplane now travels horizontally by v units.
6. If $s > 0$, decrease it by 1.

Compute how far the toy 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.

Examples

standard input	standard output
`1 1 1 1`	`1`
`2 2 2 2`	`9`
`1 2 3 4`	`68`
`314 159 265 358`	`581062`

Problem B. Business

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

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 ($1 \le l \le h \le n \le 3 \times 10^4$, $h - l \le 1000$), 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 ($-10^4 \le p \le 10^4$). 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

standard input	standard output

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

Problem C. Cookie Production

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

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 ($1 \le n, m \le 1000$), 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.

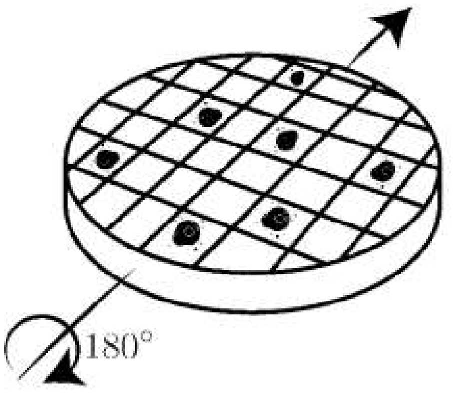
Examples

standard input	standard 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. Delicious Waffle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are using a waffle maker machine to make a delicious blueberry waffle. One side of your waffle is covered in blueberries, while the other side is plain. Initially, the cooking pan of the waffle maker lies horizontally. Once started, the cooking pan will rotate at a constant speed for a fixed duration, then stop. The cooking time is set so that when the waffle maker stops, the cooking pan will not be in a vertical position. If the cooking pan is not horizontal after this time, the waffle maker will return to a horizontal position via the smallest rotation possible. Therefore, the waffle maker will rotate less than 90 degrees, either forward or backward, until the cooking pan is horizontal again. The pan rotates at a rate of 180 degrees every r seconds, and stops after f seconds. You don't want to take out your waffle with its blueberry side down. Therefore you'd like to figure out whether the blueberry side of the waffle is up or down after the cooking pan returns to a horizontal position.



Input

The single line of input contains two integers r and f $(1 \leq r, f \leq 10^4)$. The pan rotates at a rate of 180 degrees every r seconds, and stops after f seconds. It is guaranteed that after f seconds the cooking pan is not at a vertical position.

Output

Output a single line with a single string, which is "`up`" if the blueberry side of the waffle is up, or "`down`" otherwise.

Examples

standard input	standard output
`10 20`	`up`
`10 34`	`down`
`10 47`	`down`

Problem E. Effective Management

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

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 screw- driver (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 ($1 \le n \le 3 \cdot 10^5$), which is the number of tasks you need to complete.

Each of the next n lines contains a single integer t ($0 \le t \le 2$). 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.

Examples

standard input	standard 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 F. Friends Visit

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You are a college student living on your own. However, your school friends like to visit you for several days. And they will not feel comfortable 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 friends will stop by. Since you are lazy, you want to spend as few afternoons as possible cleaning such that your friends 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 ($1 \le d \le n \le 1\,000$), where n is the number of days in your schedule and d is the number of days your friends will visit.

Each of the next n lines contains two integers m and c ($0 \le m, c \le 1\,000$). 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 ($1 \le v \le n$). These are the days on which your friends 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 friends will never see a mess. If it is not possible, output -1.

Examples

standard input	standard 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. Giant Walkway

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	1024 megabytes

Giant Walkway 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 ($2 \leq N \leq 2 \cdot 10^5$) and Q ($1 \leq Q \leq 10^5$), where is the N 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

standard input	standard output
3 3 first second third first second third first second third	0 1 0

Problem H. Hunting The Eclipses

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

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 ($0 \le d_s < y_s \le 50$), 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 ($0 \le d_m < y_m \le 50$), 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

standard input	standard output

`3 10`
`1 2`
	`7`

Problem I. Interesting Puzzle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

There is a new puzzle generating buzz on social media — Color Tubes.

The rules are relatively simple: you are given $n + 1$ tubes filled with $3n$ colored balls. Each tube can hold at most 3 balls, and each color appears on exactly 3 balls (so there are n colors).

Using a series of moves, you are supposed to reach a Color Tubes state—each tube should either hold balls of a single color or it should be empty.

The only move allowed is to take the top ball from one tube and place it into a different tube that has room for it (i.e. holds at most two balls before the move).

You want to write a program to solve this puzzle for you. Initially, you are not interested in an optimal solution, but you want your program to be good enough to solve any puzzle configuration using at most $20n$ moves.

Input

The first line of input contains a single integer n ($1 \le n \le 1\,000$), which is the number of colors.

Each of the next $n + 1$ lines contains three integers b, m and t ($0 \le b, m, t \le n$), which are the descriptions of each tube, where b is the color of the ball on the bottom, m is the color of the ball in the middle, and t is the color of the ball on the top.

The tubes are numbered from 1 to $n+1$ and are listed in order. The colors are numbered from 1 to n. The number 0 describes an empty space. It is guaranteed that no empty space will be below a colored ball.

Output

On the first line output an integer m, the number of moves that your program will use to solve the puzzle. Remember, m has to be at most $20n$.

On the next m lines, output two space-separated integers u and v that describe a move ($1 \le u, v \le n + 1$). In each move, you are taking the uppermost ball out of tube u and placing it in tube v, where it will fall until it hits the uppermost ball already in that tube, or the bottom of the tube if the tube is empty.

Your solution will be deemed incorrect if it uses more than $20n$ moves, or any of the moves are not allowed, or the final configuration is not a Color Tubes state.

Examples

standard input	standard output

```
3
2 2 0
1 3 1
3 1 2
3 0 0
```
	```
	6
	3 1
	2 3
	2 4
	3 2
	3 2
	3 4
	```

```
1
0 0 0
1 1 1
```
	`0`

Problem J. Juggle With Dice

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Given a list of three-letter words, generate one possible set of three, six-sided dice such that each word can be formed by the top faces of some arrangement of the three dice. You must distribute 18 distinct letters across the 18 total faces of the dice. There may be multiple possible sets of dice that satisfy the requirement; any correct set will be accepted.

Input

The first line of input contains an integer n ($1 \leq n \leq 1000$), which is the number of words. Each of the next n lines contains one three-letter word made up only of lowercase letters (`'a'`–`'z'`).

There may be duplicate words in the list, and the words might contain identical letters.

Output

Output a single line. If there exists a set of dice that can form all of the words, output any such set. Output the set of dice as one line with three space-separated strings, each consisting of six lowercase letters. If no such set of dice can be formed, output a single line with the number 0.

Examples

standard input	standard output
3 lad fin sly	bounds plight fakery
1 dad	0
11 aft cog far irk kit yes tau rag own uke via	vortex whacky fusing

Problem K. Kitchen

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

You have a food processor on your kitchen with a variety of blades that can be attached to it, as well as some food you would like to process into smaller pieces.

The food processor can have one blade attached at any time. Each blade processes food by reducing its average piece size at a particular exponential rate, but it also has a maximum average piece size requirement; if the average piece size of the food is too big for the blade, the food processor will get stuck. Given a starting average food piece size, a target average piece size, and a set of blades for your food processor, determine the minimum amount of processing time needed to process your food into the target average piece size.

Note that we only care about the time spent actively processing food; we do not track time spent switching out blades or loading/unloading the food processor.

Input

The first line of input contains three integers s, t, and n ($1 \le t < s \le 10^6$, $1 \le n \le 10^5$), where s is the starting average piece size, t is the target average piece size, and n is the number of blades.

Each of the next n lines contains two integers m and h ($1 \le m, h \le 10^6$). These are the blades, where m is the maximum average piece size of the blade and h is the number of seconds the blade needs to halve the average piece size.

Output

Output a single number, which is the minimum amount of time in seconds needed to process the food to the target average piece size. If it is not possible to reach the target, output -1. Your answer should have a absolute or relative error of at most 10^{-4}.

Examples

standard input	standard output

`10 1 2`
`10 10`
`4 5`
	`2.32192809488736245e+001`

`10000 9999 1`
`10000 1`
	`1.44276718045019316e-004`

Problem L. Logistics Manager

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 seconds
Memory limit:	1024 megabytes

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 ($2 \le n, m \le 2 \cdot 10^5$), 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 ($1 \le a < b \le n$), 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 ($2 \le d \le n$), 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.

Examples

standard input	standard 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. Making The Palindromes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	1024 megabytes

Given a string of lowercase English letters. You wish to make it a palindrome.

You're only a beginner at crafting of palindromes, 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.

Examples

standard input	standard output
`ioi`	`0`
`noi`	`1`
`ctsc`	`1`
`fool`	`2`
`vetted`	`2`