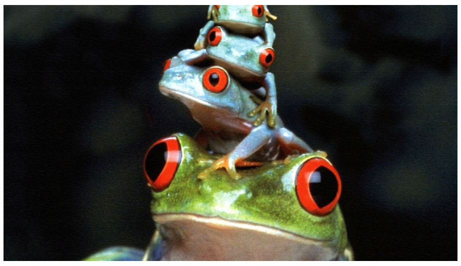
Problem A. Altruistic Amphibians

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

A set of frogs have accidentally fallen to the bottom of a large pit. Their only means of escaping the pit is to jump out of it. Each frog i is described by three parameters (l_i, w_i, h_i) where l_i is its leap capacity, w_i its weight, and h_i its height. The leap capacity specifies how high that frog can jump. If a frog's leap capacity is strictly larger than the depth of the pit, the frog can directly escape the pit. However, these frogs are altruistic. Rather than selfishly saving themselves and leaving the frogs with too limited leap capacity behind, they collectively aim to save as many of them from the pit as possible.



The frogs realize that if a frog A climbs up on the back of frog B before it jumps, the first frog A stands a better chance of escaping the pit: it can escape if $h_B + l_A$ is strictly larger than the depth of the pit.

Furthermore, if frog B carrying frog A on its back climbs up on the back of frog C, the situation is even better for frog A: it can now escape the pit if $h_C + h_B + l_A$ is strictly larger than the depth of the pit.

The frogs can build even higher piles of frogs this way, the only restriction is that no frog may carry other frogs of weight in total amounting to its own weight or heavier. Once a pile has been used to allow a frog to escape, the frogs in the pile jump back to the bottom of the pit and they can then form a new pile (possibly consisting of a different set of frogs). The question is simply how many frogs can escape the pit assuming they collaborate to maximize this number?

Input

The first line of input contains two integers n and d ($1 \leq n \leq 100\,000$, $1 \leq d \leq 10^8$), where n is the number of frogs and d is the depth of the pit in μm. Then follow n lines each containing three integers l, w, h ($1 \leq l, w, h \leq 10^8$), representing a frog with leap capacity l μm, weight w μg, and height h μm. The sum of all frogs' weights is at most 10^8 μg.

Output

Output the maximum number of frogs that can escape the pit.

Example

Input	Output

3 19
15 5 3
12 4 4
20 10 5
	3

3 19
14 5 3
12 4 4
20 10 5
	2

Problem B. Baby Bites

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

Arild just turned 1 year old, and is currently learning how to count. His favorite thing to count is how many mouthfuls he has in a meal: every time he gets a bite, he will count it by saying the number out loud.

Unfortunately, talking while having a mouthful sometimes causes Arild to mumble incomprehensibly, making it hard to know how far he has counted. Sometimes you even suspect he loses his count! You decide to write a program to determine whether Arild's counting makes sense or not.

Input

The first line of input contains an integer n ($1 \leq n \leq 1\,000$), the number of bites Arild receives. Then second line contains n space-separated words spoken by Arild, the i'th of which is either a non-negative integer a_i ($0 \leq a_i \leq 10\,000$) or the string "`mumble`".

Output

If Arild's counting might make sense, print the string "`makes sense`". Otherwise, print the string "`something is fishy`".

Example

Input	Output

`5`
`1 2 3 mumble 5`
	`makes sense`

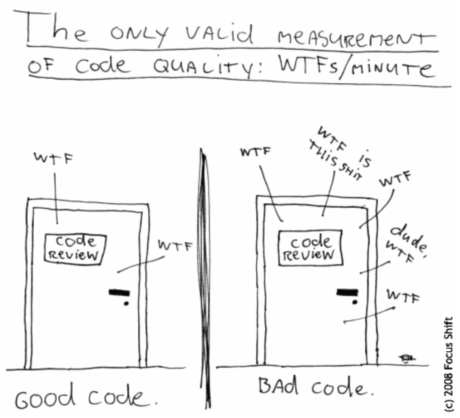
`8`
`1 2 3 mumble mumble 7 mumble 8`
	`something is fishy`

`3`
`mumble mumble mumble`
	`makes sense`

Problem C. Code Cleanups

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

The management of the software company JunkCode has recently found, much to their surprise and disappointment, that productivity has gone down since they implemented their enhanced set of coding guidelines. The idea was that all developers should make sure that every code change they push to the master branch of their software repository strictly follows the coding guidelines. After all, one of the developers, Perikles, has been doing this since long before these regulations became effective so how hard could it be?



Rather than investing a lot of time figuring out why this degradation in productivity occurred, the line manager suggests that they loosen their requirement: developers can push code that weakly violates the guidelines as long as they run cleanup phases on the code from time to time to make sure the repository is tidy.

She suggests a metric where the "dirtiness" of a developer's code is the sum of the pushes that violate the guidelines – so-called *dirty* pushes – made by that developer, each weighted by the number of days since it was pushed. The number of days since a dirty push is a step function that increases by one each midnight following the push. Hence, if a developer has made dirty pushes on days 1, 2, and 5, the dirtiness on day 6 is $5 + 4 + 1 = 10$. She suggests that a cleanup phase, completely fixing all violations of the coding guidelines, must be completed before the dirtiness reaches 20. One of the developers, Petra, senses that this rule must be obeyed not only because it is a company policy. Breaking it will also result in awkward meetings with a lot of concerned managers who all want to know why she cannot be more like Perikles? Still, she wants to run the cleanup phase as seldomly as possible, and always postpones it until it is absolutely necessary. A cleanup phase is always run at the end of the day and fixes every dirty push done up to and including that day. Since all developers are shuffled to new projects at the start of each year, no dirtiness should be left after midnight at the end of new year's eve.

Input

The first line of input contains an integer n ($1 \leq n \leq 365$), the number of dirty pushes made by Petra during a year. The second line contains n integers $d_1, d_2, \ldots, d_n$ ($1 \leq d_i \leq 365$ for each $1 \leq i \leq n$) giving the days when Petra made dirty pushes. You can assume that $d_i < d_j$ for $i < j$.

Output

Output the total number of cleanup phases needed for Petra to keep the dirtiness strictly below 20 at all times.

Example

Input	Output

`5`
`1 45 65 84 346`
	`4`

`3`
`310 330 350`
	`3`

Problem D. Delivery Delays

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

Hannah recently discovered her passion for baking pizzas, and decided to open a pizzeria in downtown Stockholm. She did this with the help of her sister, Holly, who was tasked with delivering the pizzas. Their pizzeria is an instant hit with the locals, but, sadly, the pizzeria keeps losing money. Hannah blames the guarantee they put forth when they advertised the pizzeria:

> Do you have a craving for a delicious pizza? Do you want one right now? Order at Hannah's pizzeria and we will deliver the pizza to your door. If more than 20 minutes elapse from the time you place your order until you receive your Hannah's pizza, the pizza will be *free of charge*!

Even though Holly's delivery car can hold an arbitrary number of pizzas, she has not been able to keep up with the large number of orders placed, meaning they have had to give away a number of pizzas due to late deliveries.

Trying to figure out the best way to fix the situation, Hannah has now asked you to help her do some analysis of yesterday's orders. In particular, if Holly would have known the set of orders beforehand and used an optimal delivery strategy, what is the longest a customer would have had to wait from the time they placed their order until they received their pizza?

Hannah provides you with a map of the roads and road intersections of Stockholm. She also gives you the list of yesterday's orders: order i was placed at time s_i from road intersection u_i, and the pizza for this order was out of the oven and ready to be picked up for delivery at time t_i. Hannah is very strict about following the "first come, first served" principle: if order i was placed before order j (i.e. $s_i < s_j$), then the pizza for order i will be out of the oven before the pizza for order j (i.e. $t_i < t_j$), and the pizza for order i must be delivered before the pizza for order j.

Input

The first line of input contains two integers n and m ($2 \le n \le 1\,000$, $1 \le m \le 5\,000$), where n is the number of road intersections in Stockholm and m is the number of roads. Then follow m lines, the i'th of which contains three integers u_i, v_i and d_i denoting that there is a bidirectional road between intersections u_i and v_i ($1 \le u_i, v_i \le n$, $u_i \ne v_i$), and it takes Holly's delivery car d_i time units to cross this road in either direction ($0 \le d_i \le 10^8$). There is at most one road between any pair of intersections.

Then follows a line containing an integer k, the number of orders ($1 \le k \le 1\,000$). Then follow k lines, the i'th of which contains three integers s_i, u_i, t_i denoting that an order was made at time s_i from road intersection u_i ($2 \le u_i \le n$), and that the order is ready for delivery at time t_i ($0 \le s_i \le t_i \le 10^8$). The orders are given in increasing order of when they were placed, i.e. $s_i < s_j$ and $t_i < t_j$ for all $1 \le i < j \le k$.

Hannah's pizzeria is located at road intersection 1, and Holly and her delivery car are stationed at the pizzeria at time 0. It is possible to reach any road intersection from the pizzeria.

Output

Output a single integer denoting the longest time a customer has to wait from the time they place their order until the order is delivered, assuming that Holly uses a delivery schedule minimizing this value.

Example

Input	Output

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

`3 2`
`1 2 1`
`3 2 2`
`4`
`0 3 1`
`1 3 3`
`2 2 4`
`4 3 6`
	`8`

Problem E. Explosion Exploit

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

In a two player card game, you have n minions on the board and the opponent has m minions. Each minion has a health between 1 and 6.

You are contemplating your next move. You want to play an "Explosion" spell which deals d units of damage randomly distributed across all minions. The damage is dealt one unit at a time to some remaining minion on the board. Each living minion (including your own) has the same chance of receiving each unit of damage. When a minion receives a unit of damage, its health is decreased by one. As soon as the health of a minion reaches zero, it is immediately removed from the board, before the next damage is dealt. If there are no minions left on the board, any excess damage caused by the spell is ignored.

Given the current health of all minions, what is the probability that the Explosion will remove all of the opponent's minions? Note that it does not matter if all your own minions die in the process as well, and the damage continues to be dealt even if all your own minions are gone.

Input

The first line of input contains the three integers n, m, and d ($1 \leq n, m \leq 5$, $1 \leq d \leq 100$). Then follows a line containing n integers, the current health of all your minions. Finally, the third line contains m integers, the current health of all the opponent's minions. All healths are between 1 and 6 (inclusive).

Output

Output the probability that the Explosion removes all the opponent's minions, accurate up to an absolute error of 10^{-6}.

Example

Input	Output

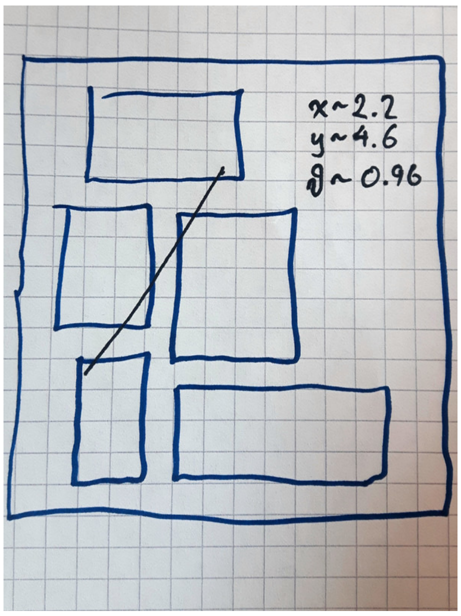
`1 2 2`
`2`
`1 1`
	`0.3333333333`

`2 3 12`
`3 2`
`4 2 3`
	`0.1377380946`

Problem F. Firing the Phaser

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

As captain of your space ship you have never encountered a more fierce enemy than the one you have snuck upon now. You immediately bring out the big phaser cannon hoping to take out the flagship before they discover you. There is no room for mistakes and the shot will have to be perfect if you are to stand any chance at all against the flagship of the enemy.



You start charging the phaser beam and retrieve the room layout of the flagship from the archives. You are situated directly above the enemy, from where the layout of the flagship can be modeled by a two-dimensional map of the rooms of the flagship. In this map, each room is a rectangle with sides parallel to the x and y axes (rectilinear), and no two rooms intersect (not even in a single point).

The phaser beam is configured by giving a point (x, y) and an angle ϑ. The phaser beam will start at (x, y) and travel a distance ℓ in the direction specified by ϑ, causing severe damage to every room touched by the phaser beam. Due to this, you aim at hitting as many rooms as possible.

The phaser beam is almost fully charged and the only missing piece is an optimal configuration of the weapon. Unfortunately, it turns out to be harder than you expected. However, there are still ten seconds before the charging is completed and hence you decide to make a computer program to solve the problem.

Input

The first line of input consists of two integers r and ℓ ($1 \le r \le 15$, $1 \le \ell \le 1\,000$) where r is the number of rooms in the flagship and ℓ is the length of a shot of the phaser.

Then follow r lines, each of which contains four integers x_1, y_1, x_2, y_2 ($0 \le x_1 < x_2 \le 1\,000$, $0 \le y_1 < y_2 \le 1\,000$), indicating that there is a room in the flagship with lower left corner (x_1, y_1) and upper right corner (x_2, y_2).

Output

Output one line with the maximum number of rooms that can be hit by one phaser beam. Recall that if the beam touches a room it is counted as a hit.

You may assume that the answer is numerically stable in the following sense: if all rooms are expanded by a distance of 10^{-6} in all four directions, the answer does not change.

Example

Input	Output
5 8	4
2 1 4 5	
5 1 12 4	
5 5 9 10	
1 6 4 10	
2 11 7 14	
3 6	3
2 2 3 3	
5 3 6 4	
6 6 7 7	

Problem G. Window on the Wall

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

Anya (Arup's daughter) would like to add a window on the wall in her room. She asks Travis (the mathematician) to figure out the largest window she can have on her wall. Travis consults Chris (the engineer) to see if there are any structural constraints. Chris explains that there must be a minimum distance between the wall perimeter and the window perimeter for the wall to hold the window; otherwise the entire structure collapses.

Given the width and height of a rectangular wall and the window-border gap (minimum distance required between the perimeter of the wall and the perimeter of the window), determine the area of the largest rectangular window that can be installed on the wall.

Input

The input contains one line with three space-separated positive integers, w, h, and d ($w, h < 1000$, $d < 100$), representing, respectively, the wall's width, wall's height, and the minimum windowborder gap amount needed.

Output

The output should be an integer on one line by itself, which represents the area of the largest rectangular window that can be installed. If it is not possible to install a window, output 0 (zero).

Example

Input	Output
40 25 5	450
30 20 12	0
30 20 50	0
999 888 7	860890

Problem H. House Lawn

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

You have just bought a new house, and it has a huge, beautiful lawn. A lawn that needs cutting. Several times. Every week. The whole summer.

After pushing the lawnmower around the lawn during the hottest Saturday afternoon in history, you decided that there must be a better way. And then you saw the ads for the new robotic lawnmovers. But which one should you buy? They all have different cutting speeds, cutting times and recharge times, not to mention different prices!

According to the advertisement, a robotic lawnmover will spend all its time either cutting the lawn or recharging its battery. Starting from a full battery, it will cut the lawn at a given rate of c square meters per minute for a cutting time of t minutes, after which it has run out of battery. Once out of battery, it will immediately start recharging. After recharging for r minutes the battery is full again and it immediately starts cutting.

You decide that in order for your lawn to look sufficiently prim and proper, the lawnmower that you buy must be powerful enough to cut your whole lawn at least once a week *on average*. Formally, if we start the mower fully charged at the beginning of the week and run it for exactly T weeks, it needs to cut the whole lawn at least T times, for all positive integers T. But apart from this, you have no specific requirements, so among the ones that satisfy this requirement, you will simply go for the cheapest option. For the purposes of cutting your lawn, you may make the simplifying assumption that a week is always exactly 10 080 minutes long.

Input

he first line of input contains two integers ℓ and m ($1 \leq \ell \leq 10^6$, $1 \leq m \leq 100$), the size of your lawn in square meters, and the number of lawnmowers to consider, respectively.

Then follow m lines, each containing a string n and 4 integers p, c, t, and r, separated by commas, describing a lawnmower as follows:

- n is the name of the lawnmower, a string of at most 60 printable characters (ASCII 32 to 126) excluding '`,`', neither starting nor ending with a space,
- $1 \leq p \leq 100\,000$ is the price of the lawnmover,
- $1 \leq c \leq 100$ is the cutting rate in square meters per minute,
- $1 \leq t \leq 10\,080$ is the cutting time in minutes, and
- $1 \leq r \leq 10\,080$ is the recharge time in minutes.

Output

Output the name of the cheapest lawnmower capable of cutting your whole yard at least once a week on average. If several lawnmovers share the same lowest price, output all of their names, in the same order they were given in the input. If there is no such mower, output "`no such mower`".

Example

Input	Output
7000 4 Grass Slayer 2000,9999,10,120,120 Slow-Mowe,999,1,120,240 Eco-cut X2,5499,2,25,35 Mowepower,5499,3,25,35	Eco-cut X2 Mowepower
100000 4 Grass Slayer 2000,9999,10,120,120 Slow-Mowe,999,1,120,240 Eco-cut X2,5499,2,25,35 Mowepower,5499,3,25,35	no such mower

Problem I. Parity of Strings

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

The historical battle between numbers and strings has taken a new twist: numbers are bragging on their categorization of being "even" or "odd" and strings lacking such feature. But don't count strings out yet!

A string is considered "even" if every letter in the string appears an even number of times; the string is "odd" if every letter in the string appears an odd number of times.

Given a string, determine whether the string is even, odd, or neither.

Input

The input consists of a single line, starting in column 1, not exceeding column 70, and containing only the lowercase letters (at least one letter).

Output

The output consists of a single integer: print 0 (zero) if the string is even, 1 (one) if the string is odd, or 2 if the string is not even and is not odd (i.e., it is neither).

Example

Input	Output
`coachessoaehwwwwww`	`0`
`coachesarefun`	`2`
`coachesc`	`1`

Problem J. Historical TV Remote Control

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

As Dr. Orooji is getting older, he is becoming more attached to older items and has difficulty letting go of them (he claims they have historical value). For example, he still has the first table he got for the programming team! The situation is the same at home, e.g., there is a broken TV remote control but Dr. O still uses it, because he considers it an old item with historical value!

The old remote control has 12 buttons: digits 0-9, channel down, and channel up. There are no other buttons on the remote control. Some digits on the remote don't work but channel up/down always works. So, to get to a particular channel, Dr. O sometimes has to use the channel up/down. For example, let's assume digits 0 and 5 on the remote don't work: If Dr. O wants to watch channel 102, he would select 99 and then "channel up" 3 times. If he wants to watch channel 597, he would select 611 and then "channel down" 14 times.

Given the digits that do not work and a target channel, determine how many times Dr. O needs to hit channel up or down. Dr. O, of course, wants to exert the least energy, hence he wants to hit the channel up/down the minimum number of times. Assume that Dr. O will enter a channel between 0 and 999 (inclusive) to start and that channel down has no effect at 0 and channel up has no effect at 999.

Input

The first input line contains an integer, n ($1 \leq n \leq 9$), indicating how many digits on the remote do not work. These broken digits are listed (in increasing order) on the same input line. The second input line provides the target channel (an integer between 1 and 999, inclusive).

Output

The output consists of a single integer, indicating how many times Dr. O needs to hit channel up/down. Note that, since one or more digits work, it is always possible to reach the target channel.

Example

Input	Output

`3 0 8 9`
`35`
	`0`

`4 1 2 5 9`
`250`
	`50`

Problem K. King's Colors

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

Far, far away, there is the mystical Kingdom of Trees (more formally, "Royal Commonwealth of Connected Undirected Simple Acyclic Graphs"). The King there is very sad because his kingdom is not accepted as a sovereign state in the United Nations. In order to become a member, he needs to make a flag the UN can put on their website.

The flag will of course consist of the King's favorite tree, which contains n nodes. The King would be happy to just have the tree colored in black and white, but for the sake of his wife the Queen, he decided that the tree will contain all the favorite colors of their k children (they all have distinct favorite colors). Clearly, no two neighboring nodes can have the same color, but otherwise any coloring of the tree using exactly the k colors would make a feasible flag candidate.

How many different flags are possible?

Input

The first line contains two integers n and k ($2 \leq k \leq n \leq 2\,500$), where n is the number of nodes in the King's favorite tree and k is the number of children. Then follow $n-1$ lines describing the edges in the tree; the i'th of these lines contains a non-negative integer $p_i < i$, meaning that node p_i is the parent of i.

The nodes are numbered from 0 to $n-1$ and the tree is rooted at node 0. Note that the embedding of the tree on the flag is already fixed, the only thing that remains is to assign colors.

Output

Output the number of different possible color assignments. The number can be quite big, so the King has requested to know the answer modulo 1 000 000 007.

Example

Input	Output

4 3
0
1
1
	18

6 4
0
1
1
3
4
	600

Problem L. First Last Sorting

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

Arup has just created a data structure that makes the two following list transformations in constant $O(1)$ time:

- Take any element in the list and move it to the front.
- Take any element in the list and move it to the back.

You've realized that sorting speed can be improved using these transformations. For example, consider the input list:

8, 3, 6, 7, 4, 1, 5, 2

We can do the following sequence of transformations to sort this list:

8, 3, 7, 4, 1, 5, 2, 6 (move 6 to end)
8, 3, 4, 1, 5, 2, 6, 7 (move 7 to end)
2, 8, 3, 4, 1, 5, 6, 7 (move 2 to front)
1, 2, 8, 3, 4, 5, 6, 7 (move 1 to front)
1, 2, 3, 4, 5, 6, 7, 8 (move 8 to end)

You are now curious. Given an input array of distinct values, what is the fewest number of these first/last operations necessary to sort the array?

Given an initial permutation of the integers $1, 2, \ldots, n$, determine the fewest number of first/last operations necessary to get the list of values sorted in increasing order.

Input

The first line of input will contain a single positive integer, n ($n \leq 10^5$), representing the number of values to be sorted. The next n lines contain one integer each. All of these integers will be distinct values in between 1 and n (inclusive), representing the original order of the data to sort for the input case.

Output

On a line by itself, output the fewest number of first/last operations necessary to sort the input list.

Example

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