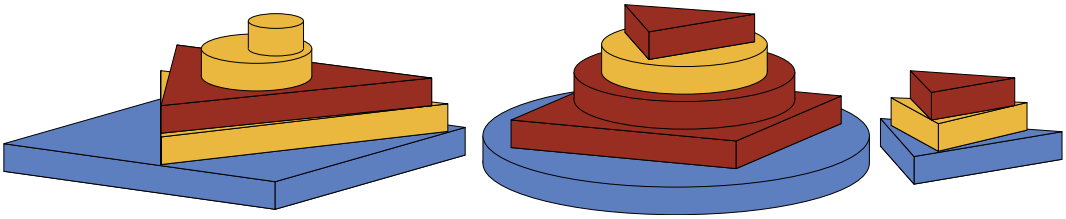
Problem A. Adolescent Architecture 2

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

Three years ago, you helped little Peter stack his toy blocks into a tower. Since then, he has extended his collection of toy blocks, which now features the following base shapes:

- `circle` a – a circle of radius a;
- `square` a – a square with side length a;
- `triangle` a – an equilateral triangle with side length a.

Here, a may be any positive integer. The top shapes of each block are the same as their bottom shapes, so the blocks are cuboids, cylinders, and triangular prisms, respectively. Peter has an infinite supply of blocks of each shape and size.



A game in progress.

Peter and his friend Amy are playing a two-player game, where the blocks need to be stacked on top of each other. Initially, some blocks are already placed on the floor. In each move, the current player must take a toy block from the infinite supply and put it on top of one of the existing stacks of blocks. The block may be rotated around its vertical axis before placing it. The outline of the new block must be strictly within the outline of the old block; the outlines are not allowed to touch. The first player who is unable to make a move loses the game.

Given the initial configuration, determine the number of winning moves for the first player.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of initial stacks.
- n lines, each with a string s (s is one of "`circle`", "`square`" or "`triangle`") and an integer a ($1 \leq a \leq 10^9$), giving the topmost blocks of the initial stacks as described above.

Output

Output the number of winning moves for the first player.

Example

Input	Output
2 circle 2 triangle 2	2
2 circle 1 circle 2	3
5 circle 123 triangle 456 square 789 square 789 triangle 555	7
3 circle 299303201 square 79724391 triangle 437068198	3
3 square 539715887 circle 518408351 triangle 348712924	0

Explanation

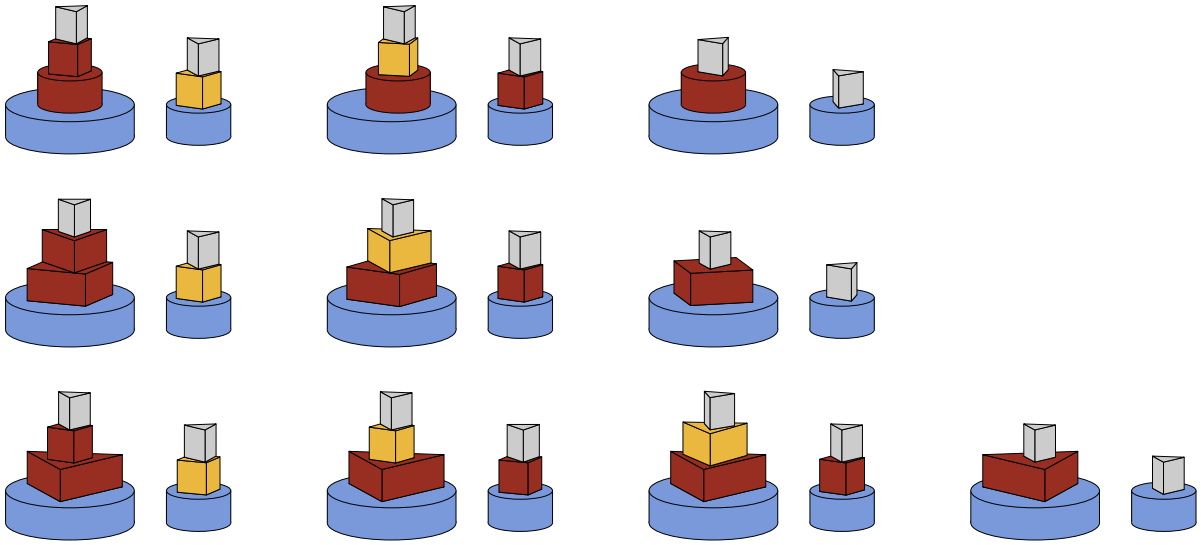


Illustration of Example Input 2, showing all possible end configurations of the game when Peter went first and played optimally to win. The blue blocks are the initial configuration. Peter needs to put one of `circle 1`, `square 2` or `triangle 3` on top of `circle 2` in order to win. Each of these options corresponds to one row of the figure. Blocks placed by Peter are coloured in red, and blocks placed by Amy are coloured in yellow. As the last two blocks are always of type `triangle 1`, they are shown in grey. If, for instance, Peter first puts `circle 1` (as depicted in the first row), then Peter can win by mirroring the following moves by Amy.

Problem B. Balloon Darts

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

As you may know, you get a colourful balloon for each problem you solve in an ICPC contest. You were quite successful in your last contest and now you own a remarkable collection of n balloons. The obvious thing to do with these balloons is to pop them all using darts. However, you only have three darts.

The balloons are modelled as points in the plane with fixed locations. For each dart you choose from where and in which direction to throw it. The dart travels in a straight line, popping all balloons in its way.

As you practised a lot during the last years, you can throw a dart precisely in any direction and it will fly infinitely far. Thus, if anyone can pop all the balloons, it is you. However, before the fun begins, you first need to determine if you can pop all balloons using at most three darts.

Popping balloons as an amusement park attraction.
Photo by blende12, Pixabay

Input

The input consists of:

- One line containing an integer n ($1 \leq n \leq 10^4$), the number of balloons.
- n lines, each containing two integers x and y ($|x|, |y| \leq 10^9$), the coordinates of a balloon.

It is guaranteed that no two balloons are at the same location.

Output

Output "`possible`" if three darts are sufficient to pop all balloons and "`impossible`" otherwise.

Example

Input	Output
6 0 0 1 1 2 4 3 9 4 16 5 25	possible
7 0 0 1 1 2 4 3 9 4 16 5 25 6 36	impossible
7 -1 -1 0 0 1 1 2 4 3 9 4 16 5 25	possible

Problem C. Cosmic Commute

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

A wormhole above Gallifrey, mau_king

A long time ago, in a galaxy far, far away, the InterCosmic Passage Company (ICPC) operates a complex railway system using *light trains*. Each planet has exactly one train station and each light train connects two distinct planets of the galaxy, going back and forth between them. Just recently, the InterCosmic Passage Company established a teleportation system, which is now in its testing phase. Some train stations are now extended by a *wormhole*. All wormholes are connected to each other, and it is possible to teleport from one wormhole to another instantaneously. To not overload the new system, each citizen of the galaxy is only allowed to teleport at most once a day.

Charlie lives on planet Gallifrey and works on planet Sontar. It is her first day of work, and she is already terribly late because her stupid alarm clock did not go off. On top of that, the new teleportation system is malfunctioning today of all days, and the destination wormhole cannot be chosen. Instead, after entering a wormhole, one is teleported to a wormhole that is chosen uniformly at random among all other wormholes. (It is impossible to be at the same train station after teleportation.)

Despite all her bad luck, Charlie is dead set on getting to work on time. Since all light trains are very slow, she wants to take as few light trains as possible. What is the expected minimum number of light trains she has to take to get to work if she can use the (malfunctioning) teleportation system at most once?

Input

The input consists of:

- One line with integers n, m, k ($2 \leq n \leq 2 \cdot 10^5$, $n - 1 \leq m \leq 10^6, 2 \leq k \leq n$), the number of planets in the galaxy, light trains and wormholes. Planet 1 is Charlie's home planet Gallifrey, and planet n is Sontar, where Charlie works.
- One line containing k distinct integers, the planets whose train stations each have a wormhole (in addition to the light trains).
- m lines, each containing two integers a and b ($1 \leq a, b \leq n$ and $a \neq b$), describing a light train between the planets a and b. It is guaranteed that all light trains are pairwise disjoint.

It is guaranteed that it is possible to travel from any planet to any other planet of the galaxy using only light trains.

Output

Output a single reduced fraction, the expected minimum number of light trains Charlie has to take to get to work if she can use the (malfunctioning) teleportation system at most once. Output the fraction as "`a/b`", where a is the numerator and b is the denominator.

Example

Input	Output
5 5 3 2 3 4 1 2 1 3 2 4 3 4 4 5	5/2
5 6 3 2 3 4 1 2 1 3 2 4 3 4 4 5 1 4	2/1

Problem D. Downsizing

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

Renowned nuclear physicist Adam Szmasczer has found a solution to global overpopulation and scarce resources: shrink everything! More precisely, he has found a way to teleport regions of space (both bounded and unbounded) to a bounded set of points inside a finite spherical cell. He explains his process as follows (from here on, we will work only in two dimensions for simplicity).

Assume a circular cell of radius r is centered at a point O. Let P be any point not in the interior of the cell, and suppose the line $\overline{OP}$ intersects the cell's boundary at the point B. Then point P is teleported to a point P' lying on this line, where $\text{length}(\overline{OP}) \cdot \text{length}(\overline{OP'}) = r^2$. (Points along the cell boundary are teleported to themselves.) Figure 3 shows how a pentagonal "house" is teleported to the shaded area within the circle; sample points O, P, P', and B are identified in the figure to illustrate the teleportation rule.

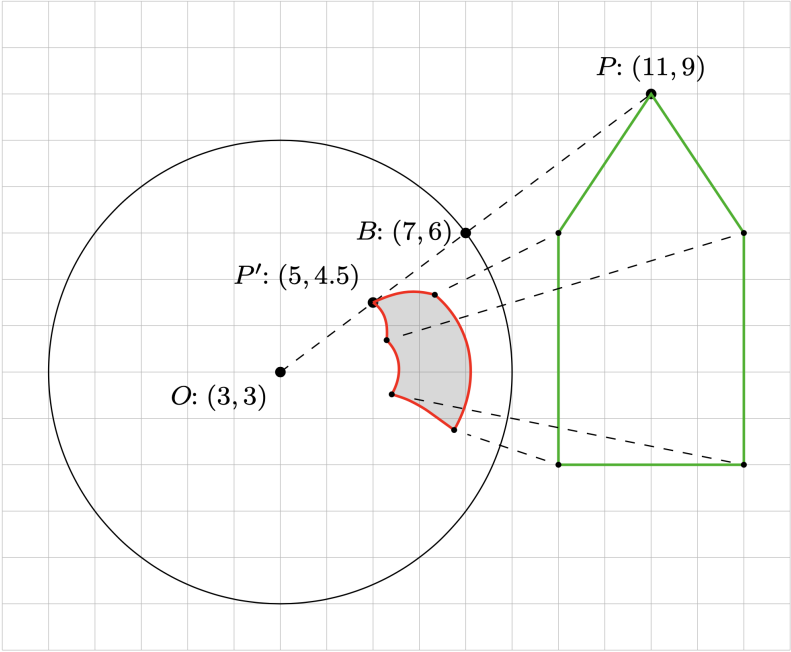


Figure 3. Example input 1, illustrating the downsizing process

Given a convex polygonal shape not containing any interior point of the cell, Adam would like to know the area of the corresponding teleported shape. Can you help?

Input

The first line of input contains three integers x_0, y_0, and r, $-10^4 \leq x_0, y_0, r \leq 10^4$, specifying the center of the circular cell and its radius. (The entire circular cell lies within the specified bounds.) The second line of input contains a positive integer n, $3 \leq n \leq 100$, followed by n pairs of integers x_1, y_1, x_2, y_2, $\ldots$, x_n, y_n, $-10^4 \leq x_i, y_i \leq 10^4$, giving the vertices of a convex polygon with n vertices in counterclockwise order.

Output

Output the area of the region of the cell corresponding to the rectangular region in the input. Your answer should be correct to a absolute error of 10^{-6}.

Example

Input	Output
3 3 5 5 11 9 9 6 9 1 13 1 13 6	4.112904

Problem E. Extended Braille

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

The Blind Association for Pretty Calligraphy is annoyed by the lack of emoticons and math symbols in the braille alphabet. Given that the braille alphabet is supported by the Unicode format, it only makes sense to make all Unicode characters supported in braille.

The goal is to extend the braille alphabet to include all Unicode characters. Of course, this will not fit in the standard 2×3 format, so using a bigger box is allowed. Important is that no two braille characters are the same up to translation, i.e., have the same shape. See Figure 4 for an example. You let a designer make up a large braille alphabet, and your job is to check how many unique shapes there are among the characters.

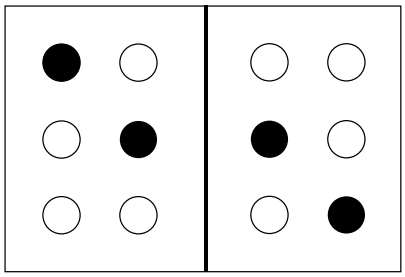


Figure 4. Illustration of Example Input 1: two characters with the same shape.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^5$), the number of braille characters.
- Then for each of the n braille characters:
  - One line with an integer m ($1 \leq m \leq 1000$), the number of dots.
  - m lines, each with two integers x and y ($|x|, |y| \leq 1000$), the coordinates of the dots.

The total number of dots is at most 10^6.

Output

Output the number of distinct braille characters up to translation.

Example

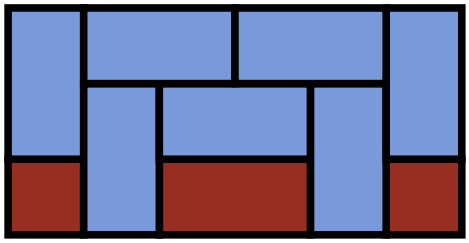
Input	Output
2 2 0 2 1 1 2 0 1 1 0	1
2 3 -1 0 0 1 1 0 3 -1 0 0 -1 1 0	2

Problem F. Freestyle Masonry

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

Fred got a simple task, he just has to build a $w \times h$ wall. To make this even easier, he was provided with enough 2×1 bricks and also a few 1×1 bricks to complete the wall. Knowing that this task should not be too hard, Fred went to work and started building the wall without thinking too much about the design. Only when he ran out of 1×1 bricks, Fred noticed that this might have been a bad idea...

An interesting brick layout, photo by Bobo Boom



Visualization of Example Input 2. The red bricks have already been placed by Fred. The blue bricks still need to be placed to complete the wall (the only possible design in this case).

Maybe he should have made a plan before starting to build the wall, but now it is too late. Fred only has a bunch of 2×1 bricks left and wants to finish the wall. Can he still complete it with the remaining 2×1 bricks? Note that the wall to be built should have a width of exactly w units and a height of exactly h units.

Input

The input consists of:

- One line with two integers w and h ($1 \leq w \leq 2 \cdot 10^5$, $1 \leq h \leq 10^6$), the width and height of the wall Fred wants to build.
- One line with w integers $h_1, \ldots, h_n$ ($0 \leq h_i \leq 10^6$), where h_i is the current height of the wall at position i.

Output

Output "`possible`" if Fred can complete his wall and "`impossible`" otherwise.

Example

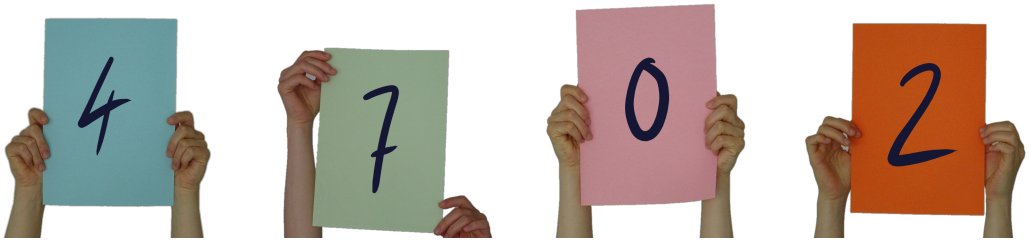
Input	Output
3 3 0 0 1	possible
6 3 1 0 1 1 0 1	possible
6 2 1 0 1 1 0 1	impossible
5 2 1 2 3 2 2	impossible

Problem G. German Conference for Public Counting

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

Greta loves counting. She practises it every day of the year. Depending on the season, she counts falling leaves, raindrops, snowflakes, or even growing leaves. However, there is one event in summer which tops everything else: the German Conference for Public Counting (GCPC).

At this event, Greta meets counting enthusiasts from all over the country for one week of counting and counting and counting... Together they participate in the Glamorous Competitive Public Counting and the Great Chaotic Public Counting. At the end of the week they all try to win the Golden Cup of Public Counting. Her favourite is the Gently Calming Public Counting where the crowd counts in silence, trying to harmoniously synchronise to reach the target number at precisely the same moment.



People holding up signs for the countdown.

To increase the tension and to prepare for the Gently Calming Public Counting, the organizers of GCPC plan to start with a silent countdown, where the people on the stage will at any time display the current number by holding up signs with its digits. On every sign, there is exactly one decimal digit. Numbers greater than 9 are displayed by holding up several signs next to each other. Each number is shown using the least possible number of signs; there is no left padding with zeroes. This way, the people on the stage display numbers $n, n-1, n-2, \ldots$ until they finally display 0. Since the GCPC will take place soon, the organizers want to finish their preparations quickly. How many signs do they need to prepare at least so that they can display the entire countdown from n to 0?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^9$), the starting number of the countdown.

Output

Output the minimum number of signs required to display every number of the countdown.

Example

Input	Output
5	6
20	11
44	14
271828182	82
314159265	82

Explanation

In the first example case, the organizers need one sign each with the digits 0 to 5, for a total of 6 signs. In the second example case, they need one sign with each digit other than 1, and two signs with a 1, for a total of $9 + 2 = 11$ signs.

Problem H. Human Pyramid

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

The Barefooted Acrobatics People's Club wants to make a group photo in an original way. For the photo, they want to make a human pyramid, where each person rests on the ground or rests on the shoulders of two people below him or her.

CC BY-SA 3.0 by Xzenia Witehira on Wikipedia

Making a human pyramid demands a lot from the acrobats involved, so the club selected a group consisting of strong people of which they are assured that these people can carry enough weight. The others are 'agile' and to make sure everyone is comfortable during the photo, there can only be agile people directly above an agile person.

The photographer wants to make a photo of a pyramid with h people on the floor, $h-1$ on the second layer, $h-2$ on the third layer, and so on, with a single person on the hth layer. You have s strong people at your disposal, and the other $\frac{1}{2}h(h+1)-s$ people are agile. What is the number of ways you can arrange the pyramid satisfying the demands of the photographer? Since this number may be large, you should find it modulo 10^9+7.

Two pyramids P_1 and P_2 are different if there exists a location where P_1 has an agile person and P_2 a strong person, or vice versa.

Input

The input consists of:

- A line containing two integers h ($1 \leq h \leq 100$) and s ($0 \leq s \leq \frac{1}{2}h(h+1)$), the number of layers in the pyramid and the number of strong people.

Output

Output the number of possible ways to build a pyramid with the given constraints, modulo 10^9+7.

Example

Input	Output
3 3	3
5 3	14

Problem I. Investigating Frog Behaviour on Lily Pad Patterns

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

A frog in a pond.

Recently, the biologist Ina discovered a new frog species on the lily pads of a pond. She observed the frogs for a while and found them to be very conscious about their personal space because they avoided sharing a lily pad with other frogs. Also, they seemed quite lazy as they did not move often, and if they did, they always jumped to the nearest empty lily pad.

To confirm her hypotheses about the frogs' movement pattern, Ina set up a large number of lily pads in a pool in her laboratory, arranged in a straight line. Since the frogs were attracted to light, she was able to simplify the test setup further by placing a bright light at one end of that line. This way, the frogs would always jump in one direction (towards the light).

Of course, Ina could now place some frogs on the lily pads and sit there all day watching the frogs jump around. But as the frogs move so rarely, it would take ages to gather a sufficient amount of data.

She therefore attached to each frog a tiny device that could log all jumps of that frog. This way, she could put the frogs on the lily pads, leave them alone for a few hours and come back later to collect the data. Unfortunately, the devices had to be so tiny that there was no space for a position tracking system; instead, the devices could only record the times of the jumps.

But if the movement pattern of the frogs is as restricted as Ina thinks, surely the individual movements of the frog can be reconstructed only from the initial positions and the recorded jump time stamps?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of frogs.
- One line with n integers $x_1, \ldots, x_n$ ($1 \leq x_i \leq 10^6$), the number of the lily pad on which the ith frog initially sits. The lily pads are numbered consecutively, starting at 1. It is guaranteed that the initial positions are strictly increasing, i.e. $x_1 < x_2 < \cdots < x_n$.
- One line with an integer q ($1 \leq q \leq 2 \cdot 10^5$), the number of jumps recorded.
- q lines, each containing an integer i ($1 \leq i \leq n$), indicating that the ith frog jumped. The jumps are given in chronological order and you may assume that a jumping frog lands before the next jump begins. The frogs always jump to the nearest empty lily pad with a larger number, and you may assume that such a lily pad always exists.

Output

For each jump, output the number of the lily pad the frog lands on.

Example

Input	Output
5 1 2 3 5 7 3 1 2 4	4 6 8
5 1 2 3 5 7 4 1 1 1 1	4 6 8 9

Explanation

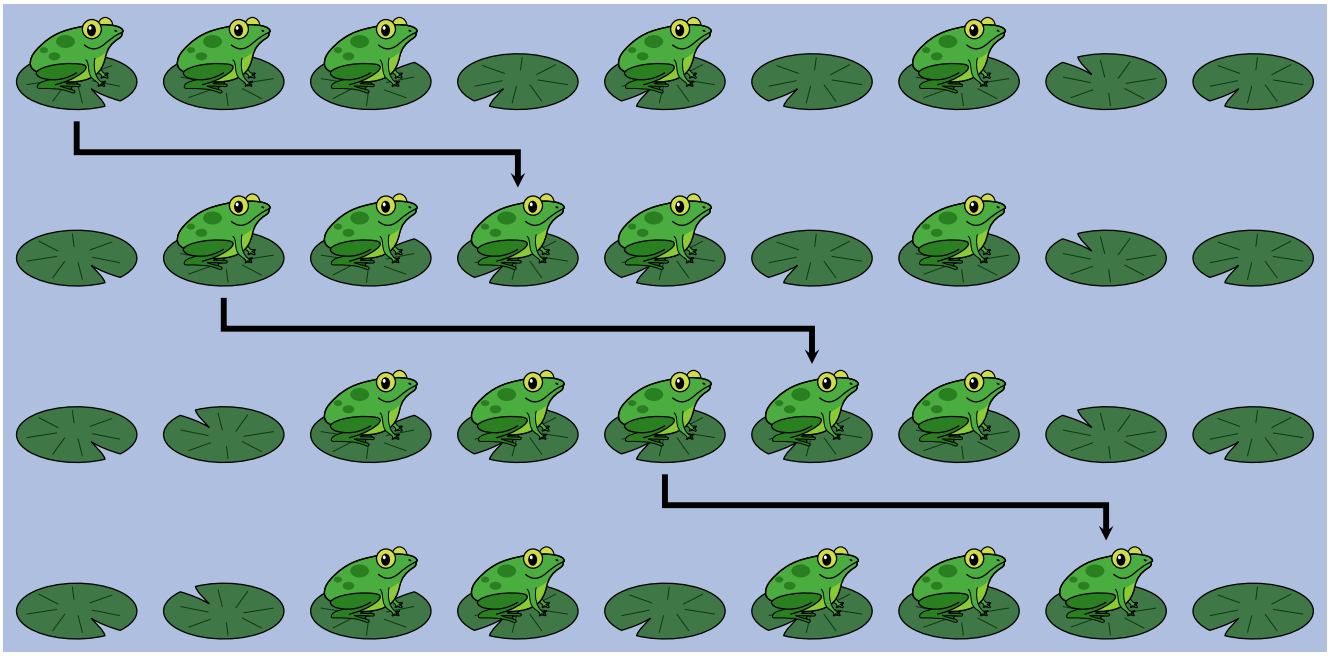


Illustration of the first example case. The lily pads are numbered from left to right, starting at 1.

Problem J. Jam-packed

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

CC-BY-SA 2.0 By treehouse1977 on Flickr

The fruit harvest has been good this year, which means that your jam-selling company, which produces the price-winning Berry Artisanal and Pure Compote, is shipping out jam left and right! A customer has recently placed a huge order of n jars of jam. To ship these jars, you put them into boxes, each of which can hold up to k jars.

As is always the case with fragile goods, the jars might break in the process of being delivered. You want to avoid the jars bouncing around in their boxes too much, as that significantly increases the chance that they break. To circumvent this, you want to avoid having boxes that are too empty: that would inevitably result in the uncontrolled bouncing around, and subsequently breaking, of the jars. In particular, you want the box with the least number of jars to be as full as possible. In order to estimate the risk you are taking with your precious jars, you would like to know: how many jars does this box contain?

Input

The input consists of:

- A line with two integers n ($1 \leq n \leq 10^{18}$), the number of jars that need to be packed, and k ($1 \leq k \leq 10^{18}$), the number of jars a single box can hold.

Output

Output the number of jars that the least filled box contains.

Example

Input	Output
10 3	2
16 4	4
1 2	1

Problem K. Kaldorian Knights

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

Painting of a medieval tournament, Codex Manesse.

The king of Kaldoria traditionally celebrates his birthday by inviting the knights of his realm to a big jousting tournament, and every noble house participates by sending their best knights to win fame and glory. At the end of the tournament, the king does not only choose a winner but ranks all n knights from worst to best.

The number of knights belonging to house i is denoted by k_i. Each knight serves at most one house. There can be knights who do not serve any house. The houses are ordered by their influence in the kingdom (with the first one being the most influential). If the k_1 knights of the most powerful house take the last k_1 places in the tournament, the house will incite a revolt against king and crown. The members of the second most influential house are not that powerful. Even if all its k_2 knights end up at the bottom of the ranking, it would be considered a strong provocation but the house would not be able to start a revolt. However, if the last $k_1 + k_2$ places are taken by all the knights of the two most influential houses combined, then the two houses will unite and start fighting the king. More generally, if the knights of the ℓ most powerful houses occupy the last $k_1 + k_2 + \cdots + k_\ell$ places in the tournament, they will unite and incite a revolt.

Of course, a revolt has to be avoided at all cost. Knowing that the king often chooses the ranking spontaneously and without too much consideration, the chief mathematician of the crown has been tasked with analysing how many rankings will *not* lead to a revolt.

Input

The input consists of:

- One line with integers n ($1 \leq n \leq 10^6$) and h ($0 \leq h \leq 5000$), the number of knights and the number of houses.
- h lines, with the ith line containing an integer k_i ($1 \leq k_i \leq n$), denoting the number of knights from house i. Note that every house is represented by at least one knight.

It holds that $\sum_{i=1}^{h} k_i \leq n$.

Output

Output the number of rankings that do not lead to a revolt. As this number can be quite large, it should be output modulo $10^9 + 7$.

Example

Input	Output
3 0	6
4 1 3	18
4 2 2 1	16

Problem L. Lottery

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

Amida-kuji is a lottery popular in Japan, which can be used to assign w prizes to w people. The game consists of w vertical lines, called *legs*, and some horizontal bars that connect adjacent legs. The tops of the legs are the starting positions of the w people, and the prizes are at the bottom of the legs. To determine the prize of the ith person, one has to move down on the ith leg, starting at the top, and switch the leg whenever a horizontal bar is encountered. You can see such a game and how to trace a path in Figure 8.

Strawberry picking game, photo by Nanao Wagatsuma

You want to manipulate the lottery in such a way that the ith person gets the ith prize, for every i, by removing some horizontal bars. Since you do not want to get caught, you want to remove as few bars as possible.

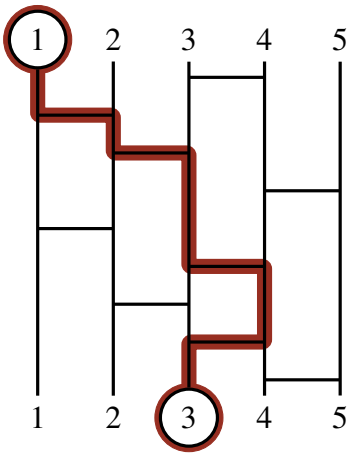


Figure 8. Visualization of an Amida-kuji game. The first person is connected to the third prize. This is also Example Input 2 after all connections are added and before any connection is removed. To connect the ith person to the ith prize, it suffices to remove both horizontal bars between legs 2 and 3 and the topmost horizontal bar between legs 3 and 4. This is the only minimal solution.

For this problem, the initial game configuration has no horizontal bars. Then, horizontal bars are added one by one or are removed again. After each change, you want to know the minimum number of horizontal bars that need to be removed such that the ith prize is assigned to the ith person for each i. Note that this is always possible by removing all horizontal bars.

Input

The input consists of:

- One line with three integers w, h and q ($2 \leq w \leq 20$, $1 \leq h, q \leq 2 \cdot 10^5$), the number of legs, the height of the legs, and the number of changes.
- q lines, each containing three integers y, x_1 and x_2 ($1 \leq y \leq h$, $1 \leq x_1, x_2 \leq w$), describing a change where a horizontal bar is added or removed at height y between legs x_1 and x_2. If there is already a horizontal bar at this position, it will be removed. Otherwise the bar will be added. It is guaranteed that the two legs are adjacent, i.e. $|x_1 - x_2| = 1$.

It is guaranteed that all horizontal bars have different heights at every moment.

Output

After each change, output a single integer, the minimum number of horizontal bars that need to be removed in the game with the currently existing bars such that the ith prize is assigned to the ith person for each i.

Example

Input	Output
4 6 7 1 1 2 2 3 4 4 3 4 5 1 2 6 3 4 3 2 3 6 3 4	1 2 1 0 1 2 1
5 9 12 1 3 4 2 1 2 3 2 3 4 4 5 5 2 1 6 4 3 7 2 3 8 4 3 9 4 5 6 4 3 7 2 3 1 3 4	1 2 3 4 3 2 3 4 3 4 3 2