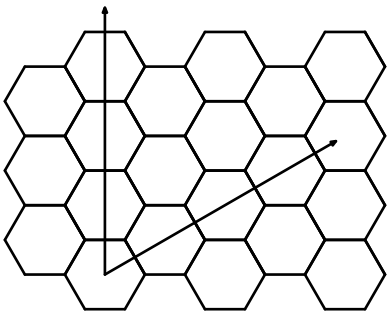
Problem A. Busy Bees

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

There is an infinite beehive consisting of hexagonal cells. Some cells contain workers which deliver honey to the queen. Each cell can contain any number of bees at any moment in time. Workers can move between cells that have a common edge, but the queen cannot move. The distance between two cells is defined as the minimum number of moves required for a worker to travel between them. The workers are very busy and they want to spend as little time as possible traveling to the queen.

You are given the coordinates of N distinct cells that contain workers. The coordinate system is illustrated below:



Find the cell where the queen should be placed so that the sum of distances between her cell and all the cells containing workers is minimized.

Input

The first line of input contains an integer N, the number of workers ($1 \le N \le 10^5$). The next N lines each contain two integers, the coordinates of the worker cells. Each coordinate does not exceed 10^9 in absolute value. It is guaranteed that all cells are distinct.

Output

Output the coordinates of the required cell. If there are multiple solutions, output any of them.

Examples

standard input	standard output
3 3 0 4 4 0 2	2 2
4 0 -2 0 0 0 2 2 0	0 0

Problem B. Convex Polygon

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Consider a convex polygon. Let (i, j)-segment be the line segment connecting the i-th and j-th vertices. Let the *double oriented chord area* (i, j) be twice the area of the shape bounded by the (i, j)-segment and the edges of the polygon going in counterclockwise order from the i-th vertex to the j-th one.

You are given a convex polygon and some queries of three types. The first query type requires removing the specified vertex from the polygon. The second type requires restoring a vertex that was removed earlier. Finally, the third query type requires calculating the double oriented chord area with given i and j. Your task is to process the given queries.

Input

The first line of the input contains the only integer n ($3 \le n \le 100\,000$). The next n lines contain two integers: the coordinates of the vertices (ranging from -10^9 to 10^9). It is guaranteed that the given polygon is non-degenerate and strictly convex (that is, the polygon is convex and no three of its vertices lie on the same line). The vertices are enumerated in counterclockwise order.

The description of the polygon is followed by an integer q, the number of queries ($1 \le q \le 100\,000$). The next q lines contain one query each.

Each query starts with one of the following characters: '`-`", `+`", or `?`". The character `-`" is followed by a vertex number and means that the respective vertex should be removed. The character `+`" is followed by a vertex number and means that the respective vertex should be restored. The character '`?`" is followed by two integers i and j and means that you have to calculate the double oriented chord area (i, j).

It is guaranteed that for the first two types of queries, no query will ask to restore a vertex that is not currently removed, and no query will ask to remove a vertex that is not currently present. For the third type, i and j will never coincide and will both be present at the time of the query. It is also guaranteed that the polygon will never have less than three vertices.

Output

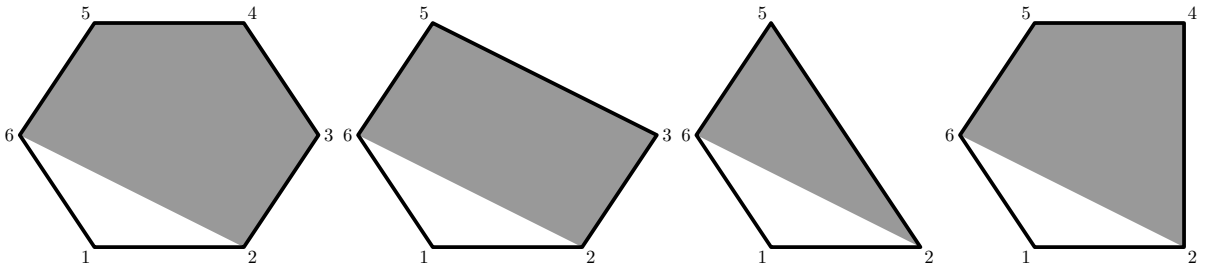
For each "`?`" query, output a single integer: the requested double oriented chord area. One can easily prove that these numbers are always integers.

Example

standard input	standard output
6 -2 -3 2 -3 4 0 2 3 -2 3 -4 0 8 ? 1 2 ? 2 6 - 4 ? 2 6 - 3 ? 2 6 + 4 ? 2 6	0 60 48 24 48

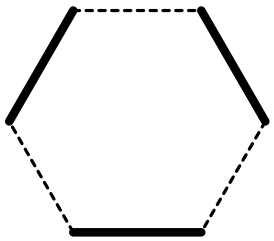
Note

Below are the states of the polygon at the time of each of the "?" queries.



Problem C. Hexagonal Billiards

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes



Consider a regular hexagon with its center at the origin $(0, 0)$ and one of its vertices at $(1, 0)$. In this hexagon, the topmost, bottom-left, and bottom-right edges have been removed, while the remaining edges are firmly fixed in their places.

You place a ball somewhere inside the hexagon, and then it starts to move with a constant speed in a uniformly distributed random direction. Each time the ball collides with one of the hexagon's edges, the collision is absolutely elastic, meaning the angle of incidence is equal to the angle of reflection, and the speed remains the same

Your task is to determine the probability of the ball leaving the hexagon after exactly N collisions.

Input

The input consists of a single line containing an integer N ($0 \le N \le 100$), followed by two real numbers x and y ($-1 < x, y < 1$) with at most two digits after the decimal point. These coordinates (x, y) represent the starting position of the ball inside the hexagon. It is guaranteed that the ball is initially located strictly inside the hexagon, with a minimum distance of 10^{-5} from each of the six initial sides of the hexagon.

Output

Output a single real number, representing the required probability, with an absolute error not exceeding 10^{-6}.

Examples

standard input	standard output
2 0.00 0.00	0.086811985
1 0.20 -0.30	0.231860631

Problem D. Circular Sequence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

A *circular sequence* of N elements is a sequence arranged in a circle such that elements 1 and 2, 2 and 3, ..., $N-1$ and N, N and 1 are adjacent. The *circular distance* between two elements i and j of the sequence is the minimal number of moves between adjacent elements to get from i to j.

You are given a circular sequence A of N integers. Your task is to select a subsequence of A that satisfies the following conditions:

1. The circular distance between any two elements of the subsequence, in their original position in A, should be at least $D+1$.
2. The sum of all numbers in the subsequence should be as large as possible.

Input

The first line of input contains two integers N and D ($1 \le N \le 100\,000$, $0 \le D < N$). The second line contains the elements of the circular sequence A separated by spaces. The elements are integers ranging from 1 to 10^9.

Output

Output a single integer: the sum of all numbers in the required subsequence.

Examples

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

Problem E. Largest Triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

Given a convex polygon, you are required to find the triangle with the largest area among all triangles whose vertices coincide with the vertices of the polygon.

Input

The input consists of no more than 50 test cases. The first line of each test case contains an integer n, representing the number of vertices of the polygon ($3 \le n \le 20,000$). Each of the next n lines contains two integers ranging from -10^9 to 10^9, representing the coordinates of the vertices. It is guaranteed that the given polygon is non-degenerate and strictly convex, meaning it is convex and no three of its vertices lie on the same line. The vertices are enumerated in counterclockwise order. The input is terminated by a line containing a single zero, which should not be processed. The tests are generated with some kind of random.

Output

For each test case, output a single positive integer, which is equal to twice the area of the required triangle. It can be easily proved that these numbers are always integers.

Example

standard input	standard output
4 -1 -1 1 -1 2 1 0 1 5 0 0 5 0 5 2 2 4 0 2 0	4 20

Problem F. Urns and Balls

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

Consider n different urns and n different balls. Initially, there is one ball in each urn.

There is a special device designed to move the balls. Using this device is simple. First, you choose a range of consecutive urns. The device then lifts all the balls from these urns. After that, you specify a destination which is another range of urns having the same length. The device then moves the lifted balls and places them in the destination urns. Each urn can contain any number of balls.

Given a sequence of movements for this device, determine where each of the balls will be placed after all these movements.

Input

The first line contains two integers n and m, representing the number of urns and the number of movements ($1 \le n \le 100\,000$, $1 \le m \le 50\,000$). Each of the next m lines contains three integers $count_i$, $from_i$ and to_i indicating thet the device simultaneously moves all balls from urn $from_i$ to urn to_i, all balls from $from_i+1$ to urn to_i+1, ..., all balls from urn $from_i+count_i-1$ to urn $to_i+count_i-1$ ($1 \le count_i, from_i, to_i \le n$, $\max(from_i, to_i) + count_i \le n+1$).

Output

Output exactly n numbers from 1 to n: the final positions of all balls. The first number represents the final position of the ball that was initially in urn 1, the second number represents the final position of the ball from urn 2, and so on.

Examples

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

Problem G. Puzznic

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Puzznic is a famous logic game where the goal is to make all puzzle pieces disappear. The game presents the player with a playing area consisting of square cells. Each cell is either a wall, an empty cell, or a puzzle piece. Puzzle pieces have types, which are shared by at least one other piece. When two or more pieces of the same type touch, they vanish. A puzzle piece is considered to be *falling* if the cell below it is either an empty cell or another puzzle piece that is also falling. Falling pieces are not considered to be touching anything. Puzzle pieces can be moved horizontally across the playing area.

The game works in cycles of three steps:

On the first step, the player can either do nothing or choose a puzzle piece and move it one position to the left or right. The piece can only be moved if it is not falling, the destination cell is empty, and there are no adjacent matching pieces which are not falling.

On the second step, the game determines which pieces to vanish. All groups of adjacent matching pieces disappear.

On the third step, all falling pieces are shifted down by one position.

Given the initial configuration of the playing area, your task is to find a sequence of moves that will make all the puzzle pieces disappear.

Input

The input contains the description of the playing area. Each character describes the type of the cell. The character "`#`" denotes a wall. A single digit from `1` to `9` determines the type of a puzzle piece. The character "`.`" denotes an empty cell.

Both the width and height of the playing area do not exceed 7. All lines have equal width. The playing area contains at least one puzzle piece. Each line contains at least one non-empty cell. It is guaranteed that initially no puzzle piece is falling and there are no adjacent matching pieces. The playing area is closed, meaning there is no path consisting of steps in the four cardinal directions which goes from the exterior of the playing area to any of the puzzle pieces.

Output

Output the sequence of user actions necessary to achieve the goal of the game. If you choose to pass the current move, output "`-`". Otherwise, output three characters separated by spaces: the direction ('`L`" is for left and '`R`" is for right), the row, and the column of the piece to move. The first row is the topmost and the first column is the leftmost.

It is guaranteed that at least one solution exists. If there are multiple solutions, you may output any one of them. The number of moves should not exceed 1000.

Examples

standard input	standard output
##### #.41# #432# #321# ##### 	L 2 3 L 2 4 L 3 3 L 3 3 L 3 4 R 4 2
###### #1.### ##.### #1..1# ##.### ######	R 2 2 - - L 4 5 R 4 2
##### #.1.# #.#.# #1.1# ##### 	L 2 3 L 4 4
#### #.1# #.## #..# #..# ##1# #### 	L 2 3 - - R 5 2

Problem H. Emergency

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

There is a square building consisting of $n \times n$ square rooms. Each room has four passages: to the north, to the south, to the east, and to the west. Some passages lead outside the building, while others connect pairs of adjacent rooms.

Your task is to draw an arrow in each room that will show which passage to follow in case of an emergency. Of course, the arrows must be drawn in such a way that, starting from any room and moving along the arrows, one can eventually exit the building.

You were confident that you knew the number of different ways to draw such arrows, but you have just discovered that some passages have been blocked. How many different ways are there to draw the arrows now?

Input

The first line of input contains two integers n and k ($1 \le n \le 10$, $0 \le k \le 10$). The next K lines describe the blocked passages. Each line contains four integers from 1 to n, which are the coordinates of two adjacent rooms between which the passage is blocked. Each blocked passage is described only once.

Output

Output a single integer: the number of different ways to draw the arrows modulo $10^9 + 7$.

Examples

standard input	standard output
2 0	192
2 4 1 1 1 2 1 1 2 1 2 2 1 2 2 2 2 1	16

Problem I. Easter Eggs

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

You are taking photos of Easter eggs. There are multiple chicken eggs in a basket, each with a unique color. You have k different colors of paint available.

You take n identical eggs from the basket and paint each one of them with one of the k colors. After painting, you arrange the eggs in a single row and capture a photo of this arrangement. You can repeat this process to create different arrangements of colors in the photos.

Now you want to determine how many different photos you can create. Since this number can be very large, you have decided to apply a set of rules to reduce the count. According to these rules, two arrangements are considered the same if you can swap the positions of egg a_i with egg b_i to transform one arrangement into another. You continue to apply more and more rules until the number of different photos becomes reasonably small. The final set of rules consists of m different rules.

Your task is to calculate the number of different photos after applying each new rule.

Input

The first line contains two integers n and k ($1 \le n, k \le 100\,000$), representing the number of eggs and the number of available paint colors, respectively. The second line contains a single integer m ($0 \le m \le 100\,000$), indicating the total number of rules applied. Each of the next m lines contains a pair of integers a_i and b_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$), representing eggs that can be swapped in each rule.

Output

Output exactly $m + 1$ numbers, where the j-th number (ranging from 0 to m) represents the number of different photos when applying the first j rules modulo $10^9 + 7$.

Examples

standard input	standard output
4 2 3 1 2 1 3 1 4	16 12 8 5
4 2 2 1 2 3 4	16 12 9

Problem J. Cranes

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

There are $m + 1$ positions located along a straight line, numbered by integers from 0 to m inclusive. The task is to move all the boxes from one position to another. You are allowed to use n cranes. Initially, all the cranes are in position 0. Each second, each crane can either stay on its current position or move to an adjacent position to the left or to the right. Each crane can either be empty or hold a single box. The time required to attach the box to the crane or to detach it is negligible.

Find the minimum possible time required to move all k boxes from position 0 to position m. Each position can contain any number of boxes. The movement of cranes and boxes does not interfere with each other.

Input

The input consists of a single line containing three integers n, m, and k ($1 \le n, m, k \le 10^5$).

Output

Output a single integer: the number of seconds required to move all k boxes from position 0 to position m.

Examples

standard input	standard output
1 10 5	90
5 10 5	10