Problem A. Range Variation Query

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 mebibytes

You have the sequence a_n. At the start a_n is initialized by the following formula:

$a_n = n^2 \bmod 12345 + n^3 \bmod 23456$.

Your task is to process queries of next two types:

- calculate the difference between maximum and minimum on the continuous subsequence $a_i, a_{i+1}, \ldots, a_j$;
- set the value j to a_i.

Input

First line of the input contains one integer k — number of queries ($1 \leqslant k \leqslant 100\,000$).

Each of next k lines contains one query. The query is described by two integers x_i and y_i.

If $x_i > 0$, then you need to calculate the difference between maximum and minimum value on the continuous subsequence $a_{x_i}, \ldots, a_{y_i}$; $1 \leqslant x_i \leqslant y_i \leqslant 100\,000$.

If $x_i < 0$, then you need to set y_i to $a_{|x_i|}$ ($-100\,000 \le x_i \le -1$, $|y_i| \le 100\,000$).

Output

For each query of first type print one integer in the new line — answer to the query.

Examples

standard input	standard output

7
1 3
2 4
-2 -100
1 5
8 9
-3 -101
2 3
	34
	68
	250
	234
	1

Problem B. Feel Good

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Bill is developing a new mathematical theory for human emotions. His recent investigations are dedicated to studying how good or bad days influent people's memories about some period of life.

A new idea Bill has recently developed assigns a non-negative integer value to each day of human life. Bill calls this number *emotional value* of the day. The greater the emotional value is, the better the day was. Bill suggests that the value of some period of human life is proportional to the sum of the emotional values of the days in the given period, multiplied by the smallest emotional value of the day in it. This schema represents that good in general period can be greatly spoiled by one very bad day.

Now Bill is planning to investigate his own life and find the period of his life that had the greatest value. Help him to do so.

Input

The first line of the input file contains n — the number of days of Bill's life he is planning to investigate ($1 \leq n \leq 10^5$). The rest of the file contains n integer numbers $a_1, a_2, \ldots a_n$ ranging from 0 to 10^6 — the emotional values of the days. Numbers are separated by spaces and/or line breaks.

Output

On the first line of the output file print the greatest value of some period of Bill's life. On the second line print two numbers l and r such that the period from l-th to r-th day of Bill's life (inclusive) has the greatest possible value. If there are multiple periods with the greatest possible value, then print any one of them.

Example

standard input	standard output
6 3 1 6 4 5 2	60 3 5

Problem C. The Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	64 mebibytes

The array consists of N elements. Your task is to tell the sum of elements of continuous subarrays of this array.

Input

First line of the input contains two integers N and K — length of the array and number of queries ($1 \leqslant N \leqslant 100\,000$), ($0 \leqslant K \leqslant 100\,000$). Each of next K lines contains one query in one of two following formats:

1. `A` i x — set x as value of i-th element ($1 \leqslant i \leqslant n$, $0 \leqslant x \leqslant 10^9$)
2. `Q` l r — find sum of elements of the continuous subarray starting at l-th element of the array and ending at r-th, $1 \leqslant l \leqslant r \leqslant n$.

Initially the array contains only zeroes.

Output

For each `Q` query print one integer — answer to this query.

Examples

standard input	standard output
5 9 A 2 2 A 3 1 A 4 2 Q 1 1 Q 2 2 Q 3 3 Q 4 4 Q 5 5 Q 1 5	0 2 1 2 0 5

Problem D. LCA - 2

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

Given a rooted tree wtih n ($1 \le n \le 100\ 000$) vertices, labeled with sequential integers between 0 and $n-1$.

Your task is answer m ($1 \le m \le 10\,000\,000$) questions about the lowest common ancestor for pair of vertices. The lowest common ancestor w of vertices u and v is a common ancestor of u and v such that distance between root and w is as big as possible.

Requests are generated by next way. The integers a_1, a_2 and x, y, z are given in the input. Values for $a_3, \ldots, a_{2m}$ are generated using formula $a_i = (x \cdot a_{i-2} + y \cdot a_{i-1} + z) \bmod n$. First query have the format $\langle a_1, a_2 \rangle$. If answer to $i-1$-th query is equal to v, then i-th query have the form $\langle (a_{2i-1} + v) \bmod n, a_{2i} \rangle$.

Input

First line of the input contains two integers n and m, the root of a tree is in vertex labeled by 0. Second line contains $n-1$ integers; i-th of those integers is label of a parent of vertex i. Third line contains three integers x, y and z; those integers are non-negative and does not exceed 10^9.

Output

Print one integer — **the sum** of answers to all queries.

Examples

standard input	standard output

3 2
0 1
2 1
1 1 0
	2

1 2
0 0
1 1 1
	0

Problem E. Add And Max At Subsegment

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Implement efficient data structure for two types of queries: addition at subsegment and maximum at subsegment.

Input

The first line contains one integer $n (1 \leqslant n \leqslant 100000)$ – size of the array.

The second line contains n integers between 0 and 100000 – elements of the array.

The third line contains one integer $m (1 \leqslant m \leqslant 30000)$ – number of queries.

Next m lines contains description of the queries. The first you are given one letter, m – for maximum, a – for addition.

m is followed by two integers l, r; in this case, you are to find maximum among elements $a_l, \ldots, a_r$.

a is followed by three integers l, r, a; in this case, you are to add a to elements $a_l, \ldots, a_r (0 \leqslant add \leqslant 100000)$. In both cases, $1 \leqslant l \leqslant r \leqslant n$.

Output

Output answers in one line separated by space.

Examples

standard input	standard output
5 2 4 3 1 5 5 m 1 3 a 2 4 100 m 1 3 a 5 5 10 m 1 5	4 104 104

Problem F. Tom and his friends

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 mebibytes

Tom and his friends paint a fence using different colors. Each of them paint some number of consecutive sections of the fence in particular color, colors may be the same for different friends. Friends color fence in order from 1st to the m-th. For each color find number of sections of this color after all queries are performed.

Input

The first line contains two integers: n ($1 \leqslant n \leqslant 10^9$) and k ($1 \leqslant k \leqslant 50000$) — number of sections and number of different colors.

The second line contains single integer m ($0 \leqslant m \leqslant 50000$) — number of friends.

Next m lines contains description of i-th friend, given as three integers c_i, l_i, r_i ($1 \leqslant c_i \leqslant k$, $1 \leqslant l_i \leqslant r_i \leqslant n$) — number of color that i-th friend used, numbers of the first and the last painted section.

Output

Output single line with k integers: i-th number should be equal to number of sections painted in the i-th color.

Examples

standard input	standard output

5 3
4
1 3 4
2 4 5
3 2 3
1 5 5
	1 1 2

5 3
3
1 1 5
2 2 4
1 3 3
	3 2 0

Problem G. Inception

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	256 mebibytes

Aliens attack city χ! The city has n buildings, each building is a point with coordinates (x_i, y_i). Aliens can conquer any square with sides parallel to coordinate axis. They want to conquer at least k buildings. But conquering of big area is tedious and time consuming, so aliens want to minimize sidelength of the square.

Please help aliens to find the smallest square, containing at least k buildings.

Input

First line of input contains two integers n $(2 \leq n \leq 50\,000)$ — number of buildings — and k $(2 \leq k \leq n)$ — minimal number of buildings aliens want to conquer. Next n lines contain coordinates of points of buildings x_i and y_i $(1 \leq x_i, y_i \leq 10\,000)$ — integer numbers, separated by space.

Output

In the first line output single integer A — minimal sidelength of the square, containing at least k points. In the second line output two integers — coordinates x and y of the bottom-left corner of the square (so, square will contain all points (x_i, y_i) that satisfy $x \leq x_i \leq x + a$ and $y \leq y_i \leq y + a$). If there are multiple solutions, output any.

Examples

standard input	standard output

3 3
1 2
2 1
3 4
	3
	1 1

Problem H. The Race with Recharging

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	256 mebibytes

The Formula-E^2 race is held on the track of length l and width h, directed from west to east. If we will choose the coordinate system such as OX axis direction is eastward, then each car will increase its x-coordinate and keep its y coordinate during the race.

The Formula-E^2 car is powered by battery with some capacity. At the start of race the battery is fully charged; at one unit of distance one unit of charge is used.

Somewhere on the track are induction power stations placed. Each station is the segment parallel to OY axis. When the car moves over the power station (inside the segment or at its ends), its battery immediately charges to the full.

In the race $w + 1$ cars are running. i-th of them $(1 \le i \le w + 1)$ starts with point with coordinates $(0, i - 1)$. The finish line is given by equation $x = l$. If somewhen the battery charge become zero and the car is not at the charging station or not on the finish, the car retires from the race.

For each car calculate minimum capacity of the battery needed for this car to finish the race.

Input

First line of the input contains three integers n, w and l $(1 \le n, w, l \le 200\,000)$ — number of power stations, width and length of the track, respectively.

Each of next n lines contain the description of the charging station. i-th $(1 \le i \le n)$ of them contains three integers x_i, a_i, b_i $(0 < x_i < l,\ 0 \le a_i < b_i \le w)$, which describe a charging station — a segment, connecting two points (x_i, a_i) and (x_i, b_i).

You may assume that no two charging stations share a common point.

Output

Print $w + 1$ integers, i-th of them denotes minimum capacity of battery needed for i-th car to finish.

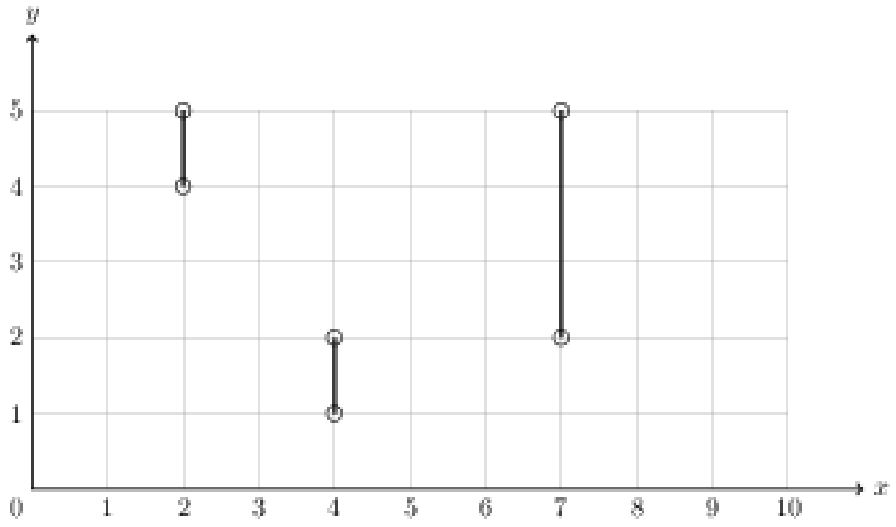
Examples

standard input	standard output

3 5 10
4 1 2
7 2 5
2 4 5
	10
	6
	4
	7
	5
	5

Note

Sample test looks like:


y
5
4
3
2
1
0
1
2
3
4
5
6
7
8
9
10
x

Problem I. K-inversions

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 megabytes

Consider a permutation $a_1, a_2, \ldots, a_n$ (all a_i are different integers in range from 1 to n). Let us call *k-inversion* a sequence of numbers $i_1, i_2, \ldots, i_k$ such that $1 \le i_1 < i_2 < \ldots < i_k \le n$ and $a_{i_1} > a_{i_2} > \ldots > a_{i_k}$. Your task is to evaluate the number of different k-inversions in a given permutation.

Input

The first line of the input contains two integers n and k ($1 \le n \le 20\,000$, $2 \le k \le 10$). The second line is filled with n numbers a_i.

Output

Output a single number — the number of k-inversions in a given permutation. The number must be taken modulo 10^9.

Example

standard input	standard output

`3 2`
`3 1 2`
	`2`

`5 3`
`5 4 3 2 1`
	`10`

Problem J. Inverse Permutation

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

The *table of inversions* for permutation $A = (a_1, a_2, \ldots, a_n)$ of integers $\{1, 2, \ldots, N\}$ is defined as array $X = (x_i)_{1 \le i \le N}$, where i-th element contains number of elemets greater than i placed at left from the i, i.e x_i = number of j', such as $j' < j$, $a_{j'} > a_j = i$.

For example, for permutation $(2, 5, 1, 3, 4)$ table of inversions looks like $(2, 0, 1, 1, 0)$, and for permutation $(6, 1, 3, 7, 5, 4, 2)$ — $(1, 5, 1, 3, 2, 0, 0)$.

The *inverse permutation* A^{-1} for permutation A is defined as permutation with the following property: i-th element of A^{-1} is equal to index of the element i in A.

For example, for permutation $(2, 5, 1, 3, 4)$ the reverse permutation is $(3, 1, 4, 5, 2)$, for permutation $(2, 7, 3, 6, 5, 1, 4)$ — permutation $(6, 1, 3, 7, 5, 4, 2)$.

Given the table of inversions for the permutation A, your task is to calculate table of inversions for inverse permutation A^{-1}.

Input

Input contains N integers — table of inversions of permutation A ($1 \le N \le 262\,144$).

Output

Print N integers — the table of inversions for A^{-1}.

Example

standard input	standard output
`2 0 1 1 0`	`1 3 0 0 0`
`5 0 1 3 2 1 0`	`1 5 1 3 2 0 0`

Problem K. Global Elephant Market

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	256 mebibytes

— Buy an elephant!
— Why would I want an elephant?
— Everyone asks, "What I need it for", just come and buy an elephant.
— Get off me!
— I will, but first you gotta buy an elephant!

This is an interactive problem.

Welcome to the world's largest elephant market. As you are a newbie to the market, your current goal is to learn how to analyze the current state of the market. You should be able to answer how much money you can make at any given moment, assuming that you have enough initial money to be able to make all the deals you wish. You will not perform any real trade.

The elephant market is based on an auction market model where a buyer bids a specific price for an elephant and a seller asks a specific price for an elephant. The prices are updated constantly as traders change their mind.

We assume that, given the current market situation, you can buy a few elephants but you must immediately sell them. You are trading just hand-to-mouth as you are not interested in keeping elephants even for a minute. So, you have to buy and sell the same number of elephants in total.

At the opening bell of the elephant market, no one wants to buy or sell elephants. Elephant offers are placed or withdrawn sequentially, listing what number of elephants one can sell or what number of elephants one can buy at specified prices. To keep things simple, you will only see a change of the number of elephants for buying or selling at the specified price. For example, "`buy 10 100`" means that you can now sell 10 more elephants for 100 piastres each and "`sell -3 99`" means that you can now buy 3 less elephants for 99 piastres each.

Write a program to determine the maximum possible profit (amount of money) you can make on the market after each offer to buy or sell elephants is placed. Note that as you do not actually perform any real trade, you must assume that after each new offer is placed, all the previous offers are still valid too.

Input

Your program will receive market offers sequentially in the following format. Every line of input corresponds to one offer and starts with the offer type (one of strings "`buy`", "`sell`" or "`end`") meaning an offer to buy some elephants, an offer to sell some elephants and the end of the trading session respectively. Offer "`end`" contains no additional parameters, and your program must exit immediately after it. Other offers contain two integers D and P separated by a single space as additional parameters describing a change D of the number of elephants on the market with appropriate order type and price P ($-10^6 \le D \le 10^6$, $1 \le P \le 10^9$). The total number of offers will never exceed 10^5.

It is guaranteed that both the total buy price of all elephants currently available on the market and the total sell price of all elephants currently available on the market will not exceed 2^{62}, and the total number of elephants for buying or selling with any price will never become negative.

Output

After each "`buy`" or "`sell`" offer, you must output a single integer on a separate line: the maximum possible profit in piastres you can make after this offer is placed on the market. Do not forget to print end-of-line characters and to flush output buffers after each output.

Examples

standard input	standard output
buy 10 100 sell 4 98 buy -7 100 buy 2 99 sell 1 97 end	0 8 6 7 9

Problem L. Chiaki With Intervals

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 mebibytes

Chiaki has a set A of n intervals, the i-th of them is $[l_i, r_i]$. She would like to know the number of such interval sets $S \subset A$: for every interval $a \in A$ which is not in S, there exists at least one interval b in S haswhich has non-empty intersection with a. As this number may be very large, Chiaki is only interested in its remainder modulo $(10^9 + 7)$.

Interval a has intersection with interval b if there exists a real number x that $l_a \leq x \leq r_a$ and $l_b \leq x \leq r_b$.

Input

The first line contains an integer n $(1 \leq n \leq 2 \times 10^5)$ – the number of intervals.

Each of the following n lines contains two integers l_i and r_i $(1 \leq l_i < r_i \leq 10^9)$ denoting the i-th interval.

It is guaranteed that for every $1 \leq i < j \leq n$, $l_i \neq l_j$ or $r_i \neq r_j$.

Output

Output an integer denoting the answer.

Examples

standard input	standard output

3
1 2
3 4
5 6
	1

3
1 4
2 4
3 4
	7

Problem M. Confusion

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	64 mebibytes

Sasha lives in a hall, and he has a lot of things. Before leaving for the summer vacation, he faced the problem of packing things. Sasha has B boxes, T types of things, N things of these types and Q operations with them. Help him to handle all transactions. There are two types of operations:

1. Find out how many things of t type are in the k-box.
2. Take all things of type t from the top n items of the k-th box and put them on the top of the m-th box.

Input

The first line contains two integers: N $(1 \le N \le 10^5)$, T $(1 \le T \le 10^5)$.
The next line contains an integer B $(1 \le B \le 10^5)$.
Each of the next B lines describes one box. It starts from an integer n $(0 \le n \le 10^5)$ - number of things in this box. Next n integers describe types of things in this box from the bottom to the top. It's guaranteed that the total number of things in all boxes equals to N.
Next line contains an integer Q $(1 \le Q \le 10^5)$.
Each of the next Q lines contains description of an operation. First of all z - type of operation. If $z = 1$ then this line contains two integers: t $(1 \le t \le T)$, k $(1 \le k \le B)$. If $z = 2$ then this line contains four integers: t $(1 \le t \le T)$, k $(1 \le k \le B)$, n $(0 \le n \le 10^5)$, m $(1 \le m \le B$, $m \ne k)$. It is guaranteed that the number of things in k-th box isn't less than n.

Output

You should output answer for all operations of the first type in separate lines.

Examples

standard input	standard output

5 2
2
3 2 2 1
2 1 2
5
1 2 1
1 2 2
2 2 1 2 2
1 2 1
1 2 2
	2
	1
	1
	2

Problem N. Mountain

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

The Mountain Amusement Park has opened a brand-new simulated roller coaster. The simulated track consists of n rails attached end-to-end with the beginning of the first rail fixed at elevation 0. Byteman, the operator, can reconfigure the track at will by adjusting the elevation change over a number of consecutive rails. The elevation change over other rails is not affected. Each time rails are adjusted, the following track is raised or lowered as necessary to connect the track while maintaining the start at elevation 0. The figure on the next page illustrates two example track reconfigurations.

Each ride is initiated by launching the car with sufficient energy to reach height h. That is, the car will continue to travel as long as the elevation of the track does not exceed h, and as long as the end of the track is not reached.

Given the record for all the day's rides and track configuration changes, compute for each ride the number of rails traversed by the car before it stops.

Internally, the simulator represents the track as a sequence of n elevation changes, one for each rail. The i-th number d_i represents the elevation change (in centimetres) over the i-th rail. Suppose that after traversing $i-1$ rails the car has reached an elevation of h centimetres. After traversing i rails the car will have reached an elevation of $h + d_i$ centimetres.

Initially the rails are horizontal; that is, $d_i = 0$ for all i. Rides and reconfigurations are interleaved through out the day. Each reconfiguration is specified by three numbers: a, b and D. The segment to be adjusted consists of rails a through b (inclusive). The elevation change over each rail in the segment is set to D. That is, $d_i = D$ for all $a \leq i \leq b$.

Each ride is specified by one number h — the maximum height that the car can reach.

Write a program that:

- reads from the standard input a sequence of interleaved reconfigurations and rides,
- for each ride computes the number of rails traversed by the car,
- writes the results to the standard output.

Input

The first line of the input contains one positive integer n — the number of rails, $1 \leq n \leq 10^9$. The following lines contain reconfigurations interleaved with rides, followed by an end marker. Each line contains one of:

- Reconfiguration — a single letter '`I`', and integers a,b and D, all separated by single spaces ($1 \leq a \leq b \leq n$, $-10^9 \leq D \leq 10^9$).
- Ride — a single letter '`Q`', and an integer h ($0 \leq h \leq 10^9$) separated by a single space;
- A single letter '`E`' — the end marker, indicating the end of the input data.

You may assume that at any moment the elevation of any point in the track is in the interval $[0, 10^9]$ centimetres. The input contains no more than 10^5 lines.

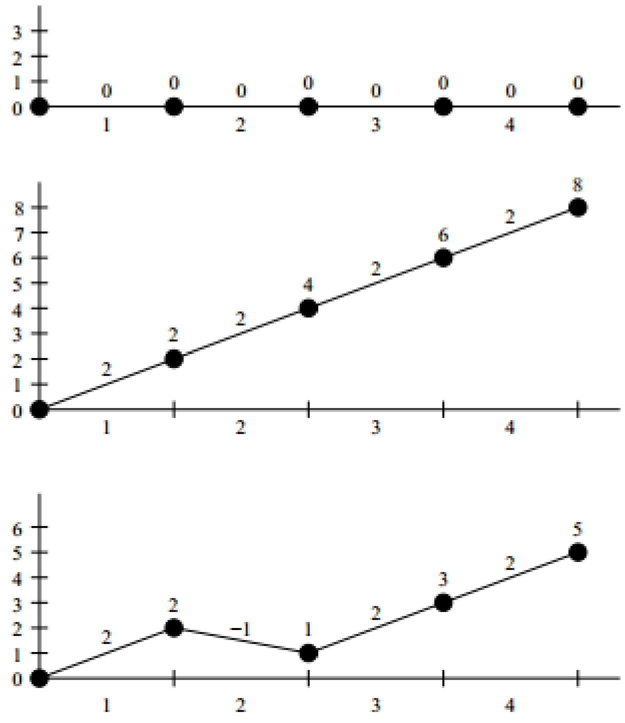
Output

The i-th line of output should consist of one integer — the number of rails traversed by the car during the i-th ride.

Example

standard input	standard output
4	4
Q 1	1
I 1 4 2	0
Q 3	3
Q 1	
I 2 2 -1	
Q 3	
E	

Note



On the pictures, you can see views of the track before and after each reconfiguration in the sample test. The x axis denotes the rail number. The y axis and the numbers over points denote elevation. The numbers over segments denote elevation changes.