Contents

In some problems input and output data is quite large. You may use extremely fast I/O functions:
Link to source files: `http://mipt2014.snarknews.info/files/mipt2014n/fastio/`

In each problem there are two Time Limits – for `C/C++` language family and for `Java/Python` language family. All other languages use TL for `C/C++`. For example, notation "5 (7.5) sec" means, `TL` for `C/C++` is 5 seconds and `TL` for `Java/Python` is 7.5 seconds.

Warm-up

Problem A. Fast addition [5 (7.5) sec, 256 mb]

There is array of integer numbers. Length of the array is $n = 2^{24}$. Initially it is filled by zeroes. You have to proceed m random queries kind of "add number to the given segment". Then you have to proceed q random queries kind of "find sum on the given segment".

Input

The first line contains numbers m, q ($1 \leqslant m, q \leqslant 2^{24}$). The second line contains pair of integer numbers a, b from 1 to 10^9. This pair is used in generator of random numbers.

```
0.  unsigned int a, b; // input data
1.  unsigned int cur = 0; // 32-bit integer
2.  unsigned int nextRand() {
3.      cur = cur * a + b; // computation with overflow
4.      return cur >> 8; // number from 0 to 2^24 - 1.
5.  }
```

Each query of the first type is generated as follows:

```
1.  add = nextRand(); // the number to add to the segment
2.  l = nextRand();
3.  r = nextRand();
4.  if (l > r) swap(l, r); // we've got the segment [l..r]
```

Each query of the second type is generated as follows:

```
1.  l = nextRand();
2.  r = nextRand();
3.  if (l > r) swap(l, r); // we've got the segment [l..r]
```

At first the first type queries are generated, then the second type queries.

Output

Output sum of all answers to all queries of the second type modulo 2^{32}.

Examples

fastadd.in	fastadd.out
5 5 13 239	811747796

Note

Sequence of queries from the sample:

```
[13..170] += 0
[28886..375523] += 2221
[2940943..13131777] += 4881801
[2025901..10480279] += 4677840
[4943766..6833065] += 9559505
get sum [13412991..13937319]
get sum [1871500..6596736]
get sum [7552290..14293694]
get sum [1268651..16492476]
get sum [2210673..13075602]
```

Problem B. All minimums [0.25 (0.5) sec, 256 mb]

You are given an array of integer numbers $a_1, a_2, \ldots, a_n$.
For each subsegment $[a_L..a_R]$ let denote $F(L, R) := min\{a_L, \ldots, a_R\}$.
Calculate

$$\sum_{1 \leqslant L \leqslant R \leqslant n} F(L, R)$$

mean "sum of minimums of all subsegments".

Input

The first line contains integer n ($1 \leqslant n \leqslant 100\,000$) — size of the array. The second line contains elements of the array separated by spaces, all numbers are integers from -10^6 to 10^6.

Output

Output the only number — sum of minimums of all subsegments of the array a.

Examples

minsum.in	minsum.out
1 5	5
2 -10 1	-19
4 1 2 3 4	20
5 -3 2 -4 1 -5	-52

The meat

Problem C. Yet another k-th statistic [5 (7.5) sec, 256 mb]

Initially you have an array of integer numbers.
You have to proceed three types of queries:

- `+ i x` — Insert the number x to the i-th position (size of the array increases by one)
- `- i` — Erase the number on i-th position of array (size of the array decreases by one)
- `? L R x` — Say, how many numbers y on positions $L \leqslant i \leqslant R$ such, that $y \leqslant x$ ($|x| \leqslant 10^9$)

All indeces i, L, R are numbered from zero. All numbers in queries are integers. All queries are correct. Example of the query: `"+ 0 x"` means "insert x to the beginning of the array". Initially amount of elements in array is N ($0 \leqslant N \leqslant 10^5$). Numbers in array do not exceed 10^9 by absolute value. Amount of queries is K ($1 \leqslant K \leqslant 10^5$).

Example

kthstat.in	kthstat.out
10 455184306 359222813 948543704 914773487 861885581 253523 770029097 193773919 581789266 457415808 - 1 ? 2 5 527021001 ? 0 5 490779085 ? 0 5 722862778 + 9 448694272 - 5 ? 1 2 285404014 - 4 ? 3 4 993634734 + 0 414639071	1 2 2 0 2

Problem D. Shots at Walls [3 (4.5) sec, 256 mb]

A new pistol is being tested. The pistol can fire shots with variant bullet speeds. In some points of time shots are fired from the point of origin with certain horizontal speeds, and in some other points of time walls are built on a horizontal platform. The walls are non-singular segments lying on lines that do not go through the point of origin. The walls may intersect. For processing of the test results, you are to determine the time that each shot bullet had been flying for. You can assume that the speed of the bullet after shot is constant. The pistol stops immediately after touching any wall.

Input

Each line of the input begins with either "shot", "wall", or "end" (without quotes). The number of lines doesn't exceed 50 000. After "shot", the two coordinates of speed of the bullet are listed; the speed cannot be zero. After "wall", the four numbers follow, being the coordinates of wall's beginning and end. "end" denotes the end of the input. All the coordinates are integers whose absolute values doesn't exceed 10 000. All the events are listed in chronological order, and time intervals between the events exceed the time needed to build a wall, or the time needed for bullet to reach the next wall or end of the proving ground.

Output

For each of the shots, you must output the single number, on a line by itself: the time that the bullet had been flying for, with precision of 10^{-6}. If the bullet doesn't hit any wall, you must output "Infinite" instead of a number.

Example

shots.in	shots.out
shot 1 0 wall 1 0 0 1 shot 1 1 shot -1 3 wall 1 0 -1 2 shot -1 3 wall 1 1 -1 1 shot -1 3 wall 2 3 2 -3 wall 3 -2 -3 -2 shot 1 -1 shot 40 -39 shot 9999 -10000 shot -1 -1 shot -3000 -2000 shot -3001 -2000 shot -3000 -2001 shot 1 0 shot 1 1 wall -1 2 10 -10 shot -1 1 shot 0 1 shot 1 1 shot 1 0 shot 1 -1 wall 0 -10000 -10000 0 shot -2 -1 end	Infinite 0.50000000000000000000 Infinite 0.50000000000000000000 0.33333333333333333333 2.00000000000000000000 0.05000000000000000000 0.00020000000000000000 2.00000000000000000000 0.00100000000000000000 Infinite 0.00099950024987506247 1.00000000000000000000 0.50000000000000000000 1.00000000000000000000 0.90909090909090909091 0.43478260869565217391 0.83333333333333333333 2.00000000000000000000 3333.33333333333333333

Problem E. The site for ROI [2 (4) sec, 64 mb]

Organizers of ROI-2239 (Russian Olympiad in Informatics) plan to hold the Olympiad in nature. For this purpose they chose small grove and plan to organize square site in it. The participants of the event will be located on this site. To Earth's magnetic field does not engage in interference with wireless Network, which is used to join participants' computers, sides of the site should be parallel to directions North-South, West-East.

However n valuable trees grows in the grove. Activists of the environmental organization "Greenpeace" insist that during the event none of the trees are not affected. In particular, they threaten, that if the organizers of the event will try to damage at least one tree, then Activists prick themselves to trees with handcuffs superalloy and will interfere with the Olympiad with the regular obscene shouts.

Therefore, the organizers are planning to organize a site in such a way, that does not damage the trees. To take the maximum number of participants, organizers want the size of the site was the maximum possible. Help them to determine the maximum size of the site which you can build and find a place where it should be built.

Input

We assume that the grove is of rectangular shape with sides parallel to the sides of the world. Organize a system of coordinates so that the axes are parallel the cardinal, the coordinates of one corner of the grove is $(0, 0)$ and the opposite corner is (x, y).

The first line contains two integers — x and y ($0 \leqslant x, y \leqslant 10^9$). The second line contains the number of trees n ($0 \leqslant n \leqslant 200\,000$). The next n lines contain the coordinates of the trees — two numbers per line x_i, y_i ($0 \leqslant x_i \leqslant x$, $0 \leqslant y_i \leqslant y$). No two trees coincide.

Output

On the first line of the output file output a single number — the maximum length of the side of square site with sides parallel to coordinate axes, wich can be built without damaging any of precious tree. The site must be completely inside the grove.

On the second line output two integers — coordinates of the corner of the site, nearest to the point $(0, 0)$. If there are several optimal answers, output lexicographically smallest one (ie, with the smallest possible x, and if there are several — with the smallest possible y).

Examples

roi.in	roi.out
3 3 1 1 1	2 0 1