Problem A: Queue sum

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

We have a queue of integer values. Every now and then, the first element gets removed and pushed back at the end. We want to calculate sums over intervals of the queue at different points in time.

Input

The input consists of

- one line containing N and M ($1 \leq N, M \leq 10^6$) – the number of elements in the queue and the number of queries
- one line containing N numbers $a_1, ..., a_N$ ($0 \leq a_i \leq 10^8$) – the elements in the queue in the beginning
- M lines each containing one letter, either `r`, signaling that the first element is removed and pushed back at the end, or `q`, indicating a sum query, and if it's `q` also two numbers l_i and r_i ($1 \leq l_i \leq r_i \leq n$), the beginning and end of the interval to sum over.

Output

For any query "`q` l r", output the sum of elements from l to r (inclusively) on a separate line.

Sample input and output

Input	Output
3 5 1 2 3 q 1 2 r q 2 3 r q 2 2	3 4 1

Problem B: Binary Tree

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Given a list $a_1, ..., a_N$ of numbers, find a binary tree with N vertices s.t.

- the in-order traversal (i.e. the traversal scheme that, when deployed in a binary search tree with unique values, retrieves the data ascendingly sorted) of the tree visits the vertices in the order $1, ..., N$
- for every vertex i the depth (distance from root) of i is a_i.

Input

The input consists of

- one line containing N ($1 \leq N \leq 10^5$) – the number of vertices
- one line containing the depth values $a_1, ..., a_N$ ($0 \leq a_i < N$).

It is guaranteed that for every set of parameters you are given there is at least one tree that conforms to the constraints stated above.

Output

Output a single line containing N integers, the i-th being the parent of vertex i for all vertices except the root, and 0 for the root.

Sample input and output

Input	**Output**
3 2 1 0	2 3 0
5 2 1 2 0 1	2 4 2 0 4

Problem C: Classroom

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Due to massive budget cuts in his state's education budget, high school teacher Mr. Gregarius is teaching an enormously large class this term. Since in such a setting it is generally pretty hard for *any* teacher to keep the class quiet and get everyone's attention, he always wanders around the classroom aimlessly (to at least be *seen* by most students during a lesson), frequently poses a difficult question and, seemingly at random, picks a nearby student to answer it.

But of course he doesn't *actually* want to pick students at random! Over the course of the first few lessons he numerated the students 1,...,N and successfully determined each student's *general performance index* (GPI), which is a nonnegative integer that is greater, the better a student performs. Now whenever he has posed a question, he only restricts himself to a certain range of currently nearby students, then tracks down the lowest performing one in that range and asks him to answer it.

Since the class is so large, Mr. Gregarius is having trouble keeping everyone's GPI in mind, therefore he asked you to write him a little program that continuously runs throughout the term, keeps track of changes in the class structure and quickly tells him for any given range of students which is the lowest performing one in that range. Specifically, the program should be able to process queries of the following kinds:

- `QUESTION` l r
  signaling that he just asked a question and now wants to know the index of the lowest performing student within the range $l, ..., r$ of students
- `DROPOUT` i
  signaling that the student with index i dropped out of school and will no longer be present to answer any questions

Input

The input consists of

- one line containing N ($3 \leq N \leq 10^5$) – the number of students in the class (in the beginning of the program's execution, before anyone has dropped out)
- one line containing the GPIs $g_1, ..., g_N$ ($0 \leq g_i \leq 10^8$, g_i is the GPI of the student with index i). It is guaranteed that no two students share a GPI and that Mr. Gregarius only picks ranges that still include at least one student.
- many lines each containing a query of one of the types described above. The program is supposed to terminate when given the query `DROPOUT` -1. It is guaranteed that there will be no more than 10^5 queries in a single execution of the program.

Output

For every query of the `QUESTION` type, output the requested student index on a separate line.

Sample input and output

Input	**Output**
5 42 7 19 13 0 QUESTION 1 3 QUESTION 4 5 DROPOUT 5 QUESTION 3 5 DROPOUT -1	2 5 4

Problem D: Rainfall

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

The National Weather Service wants to provide a new online tool in which users can look up the projected rainfall for their location (a cell on an $M \times N$ grid that models the country) by a single request. Since the meteorologists working there can't code, they hired you to help them by writing a program that can quickly process large amounts of queries of the forms

- `RAIN` x_1 y_1 x_2 y_2 r
  denoting a projected rainfall of r mm on the area $[x_1, x_2]_{\mathbb{N}} \times [y_1, y_2]_{\mathbb{N}}$ (i.e. on all cells (x, y) where $x_1 \leq x \leq x_2, y_1 \leq y \leq y_2$)
- `REQUEST` x y
  denoting a request of the currently projected rainfall in location (x, y)

s.t. they can run it every day to give their users forecasts for tomorrow.

Input

The input consists of

- one line containing N and M ($3 \leq N, M \leq 1000$) – the width and height of the grid
- many lines, each containing a query of one of the types described above (all numbers involved are integers). The program is supposed to terminate when given the query "`REQUEST` -1 -1". It is guaranteed that there will be no more then 10^5 queries in a single execution of the program and that all but the terminating query will only contain valid coordinates (i.e. ones ranging from 1 to N in $x-$ and from 1 to M in y-direction) and that the `RAIN` queries always describe positive amounts of projected rainfall, not exceeding 100 per cell, on non-empty areas.

Output

For every non-terminating query of the `REQUEST` type, output the correct result (in mm) to that request on a separate line.

Sample input and output

Input	Output
3 3 RAIN 2 1 3 1 5 RAIN 1 1 2 2 15 REQUEST 2 1 RAIN 2 2 3 3 25 REQUEST 1 3 REQUEST 2 2 RAIN 1 1 3 3 100 REQUEST -1 -1	20 0 40

Problem E: Enemy Ships

Advanced Algorithms for Programming Contests

Restrictions

Time: 4 seconds
Memory: 1.2 GB

Problem description

The Milky Way is under attack by its closest neighbor, the Andromeda Galaxy. Thousands of andromedan space ships are invading, and the Milky Way's galactic council seems utterly overwhelmed by the situation. To give some strategic basis to their defense, and to enable trade ships to (relatively) safely continue their travels through the Milky Way, they have requested a program that keeps track of when and where enemy ships have been sighted.

The 2D map of the Milky Way they want to work with divides the galaxy into a 500×500 grid of quadratic cells, each of which denotes a *galactic sector*. The program should act on a timescale of days, reaching from 0 to 1000, and it should be able to quickly and adequately respond to queries of the following forms:

- t `SIGHTING` n x y
  signaling a sighting of n enemy ships in sector (x, y) on day t

- t `TRAVEL` p n x y r
  signaling that on day t a trade ship requests whether sector (x, y) is safe enough for it to go there – specifically, whether in the past p days (up to and including the current day t, so in days $t - p + 1, ..., t$) less than n enemy ships were sighted in the r-cube of sectors around (x, y) (i.e. in $[x - r, x + r] \times [y - r, y + r]$, where $r \geq 0$ is guaranteed); the output should be `YES` if it is safe and `NO` otherwise

- t `COUNTER` p x_1 y_1 x_2 y_2
  signaling that on day t a part of the Milky Way's defense troops is considering a counterattack that would disperse andromedan invaders from all sectors in the area $[x_1, x_2] \times [y_1, y_2]$ and wants to know exactly how many enemy ships have been sighted in said area in the past p days (see above)

Often times, the program will be asked to give out information about sightings on the current day (i.e. the day the request is made) long before all

sightings of that day have been registered by it. This problem is to be disregarded (i.e. the program should always reply instantly, based solely on the information it received up to the request).

Input

The input consists entirely of queries of the types described above, each printed in a separate line. All coordinates – including all corners of r-boxes in `TRAVEL` queries – are guaranteed to be integers lying on the map (reaching from 1 to 500 in both coordinate directions) and all query timestamps t are guaranteed to lie between 1 and 1000 (incl.). Also, no request will ask about sightings on days with indices smaller than 1 (i.e. $t - p + 1 \geq 1$), sightings will only ever report positive amounts of enemy ships not exceeding 1000 per sighting, and – of course – the queries will be sorted by the times t at which they are issued. The program is supposed to terminate when given the query

$$-1 \texttt{ SIGHTING } -1\ -1\ -1.$$

You may assume that there will be no more than 10^5 queries in a single execution of the program.

Output

For every non-terminating query of the types `TRAVEL` and `COUNTER`, output the correct result (as specified above) to that query on a separate line (in the order in which the queries are received).

Sample input and output

Input	**Output**
13 SIGHTING 332 5 8 22 TRAVEL 10 5 6 6 2 23 TRAVEL 10 5 6 6 2 37 SIGHTING 68 117 12 37 COUNTER 30 1 1 120 12 -1 SIGHTING -1 -1 -1	NO YES 400

Problem F: Impacts

Advanced Algorithms for Programming Contests

Restrictions

Time: 2 seconds
Memory: 512 MB

Problem description

Impacts of objects on the moon are of high scientific interest. There are several sensor stations on the moon surface that measure the vibrations caused by such impacts. You get better data, the closer the impact is to a sensor station. We already know, when and where impacts occurred and want to find out for given time windows, what the minimal distance of an impact in that window to any of the sensor stations is.

Input

The input consists of

- one line containing k, n, q ($1 \leq k \leq 20$, $0 \leq n \leq 10^5$, $1 \leq q \leq 10^5$) – the numbers of sensor stations, impacts and time window queries
- k lines each containing two integers $x_i^{\text{sta}}, y_i^{\text{sta}}$ ($|x_i^{\text{sta}}|, |y_i^{\text{sta}}| \leq 1000$) – the coordinates of the i-th sensor station
- n lines each containing three integers $x_i^{\text{imp}}, y_i^{\text{imp}}, t_i^{\text{imp}}$ ($0 \leq t_i^{\text{imp}} \leq 10^8$, $|x_i^{\text{imp}}|, |y_i^{\text{imp}}| \leq 1000$), the coordinates and time of the i-th impact. The impacts are given in correct time order. The next q lines each contain two integers b_i and e_i ($0 \leq b_i \leq e_i \leq 10^8$) – beginning and end of the i-th time window.

Output

For every query "b_i, e_i" output the closest distance of any impact not earlier than b_i and not later than e_i (in $[b_i, e_i]$) with a precision of at least 10^{-6}. If there was no impact in that time window, output "INF".

Sample input and output

Input	Output
2 3 4 0 0 5 2 5 4 2 0 4 4 -1 -1 5 0 1 2 3 3 4 1 6	INF 2.0000000 4.0000000 1.41421356