# Points

time limit per test

2 seconds

memory limit per test

256 megabytes

Pete and Bob invented a new interesting game. Bob takes a sheet of paper and locates a Cartesian coordinate system on it as follows: point (0, 0) is located in the bottom-left corner, *Ox* axis is directed right, *Oy* axis is directed up. Pete gives Bob requests of three types:

* add x y — on the sheet of paper Bob marks a point with coordinates (*x*, *y*). For each request of this type it's guaranteed that point (*x*, *y*) is not yet marked on Bob's sheet at the time of the request.
* remove x y — on the sheet of paper Bob erases the previously marked point with coordinates (*x*, *y*). For each request of this type it's guaranteed that point (*x*, *y*) is already marked on Bob's sheet at the time of the request.
* find x y — on the sheet of paper Bob finds all the marked points, lying strictly above and strictly to the right of point (*x*, *y*). Among these points Bob chooses the leftmost one, if it is not unique, he chooses the bottommost one, and gives its coordinates to Pete.

Bob managed to answer the requests, when they were 10, 100 or 1000, but when their amount grew up to 2·105, Bob failed to cope. Now he needs a program that will answer all Pete's requests. Help Bob, please!

**Input**

The first input line contains number *n* (1 ≤ *n* ≤ 2·105) — amount of requests. Then there follow *n* lines — descriptions of the requests. add x y describes the request to add a point, remove x y — the request to erase a point, find x y — the request to find the bottom-left point. All the coordinates in the input file are non-negative and don't exceed 109.

**Output**

For each request of type find x y output in a separate line the answer to it — coordinates of the bottommost among the leftmost marked points, lying strictly above and to the right of point (*x*, *y*). If there are no points strictly above and to the right of point (*x*, *y*), output -1.

**Examples**

**input**

**Copy**

7  
add 1 1  
add 3 4  
find 0 0  
remove 1 1  
find 0 0  
add 1 1  
find 0 0

**output**

**Copy**

1 1  
3 4  
1 1

**input**

**Copy**

13  
add 5 5  
add 5 6  
add 5 7  
add 6 5  
add 6 6  
add 6 7  
add 7 5  
add 7 6  
add 7 7  
find 6 6  
remove 7 7  
find 6 6  
find 4 4

**output**

**Copy**

7 7  
-1  
5 5

题意：在一个坐标系的第一象限内有三个操作：

1、add x y；标记(x, y)点；

2、remove x y；消除已经标记过的点(x, y)

3、find x y；在点(x, y)的右上方（严格）找出在横坐标方向上离(x, y)最近的点，若有多个满足条件的点，就找纵坐标最小的点，找到后输出该点坐标；如果没有就输出-1

# 52C. Circular RMQ

time limit per test

1.5 seconds

memory limit per test

256 megabytes

You are given circular array *a*0, *a*1, ..., *an*- 1. There are two types of operations with it:

* *inc*(*lf*, *rg*, *v*) — this operation increases each element on the segment [*lf*, *rg*] (inclusively) by *v*;
* *rmq*(*lf*, *rg*) — this operation returns minimal value on the segment [*lf*, *rg*] (inclusively).

Assume segments to be circular, so if *n* = 5 and *lf* = 3, *rg* = 1, it means the index sequence: 3, 4, 0, 1.

Write program to process given sequence of operations.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 200000). The next line contains initial state of the array: *a*0, *a*1, ..., *an*- 1 ( - 106 ≤ *ai* ≤ 106), *ai* are integer. The third line contains integer *m* (0 ≤ *m* ≤ 200000), *m* — the number of operartons. Next *m* lines contain one operation each. If line contains two integer *lf*, *rg* (0 ≤ *lf*, *rg* ≤ *n* - 1) it means *rmq* operation, it contains three integers *lf*, *rg*, *v* (0 ≤ *lf*, *rg* ≤ *n* - 1; - 106 ≤ *v* ≤ 106) — *inc* operation.

**Output**

For each *rmq* operation write result for it. Please, do not use %lld specificator to read or write 64-bit integers in C++. It is preffered to use cout (also you may use %I64d).

**Examples**

**Input**

**Copy**

4  
1 2 3 4  
4  
3 0  
3 0 -1  
0 1  
2 1

**Output**

**Copy**

1  
0  
0

题意：对于一个给定的环形数组，有两种类型的操作：

1、inc(lf, rg, v)，给区间[lf, rg]中的每个元素增加v

2、rmq(lf, rg)，求区间[lf, rg]中的最小值。