

D. Welfare State

time limit per test: 2 seconds
 memory limit per test: 256 megabytes
 input: standard input
 output: standard output

There is a country with n citizens. The i -th of them initially has a_i money. The government strictly controls the wealth of its citizens. Whenever a citizen makes a purchase or earns some money, they must send a receipt to the social services mentioning the amount of money they currently have.

Sometimes the government makes payouts to the poor: all citizens who have strictly less money than x are paid accordingly so that after the payout they have exactly x money. In this case the citizens don't send a receipt.

You know the initial wealth of every citizen and the log of all events: receipts and payouts. Restore the amount of money each citizen has after all events.

Input

The first line contains a single integer n ($1 \leq n \leq 2 \cdot 10^5$) — the number of citizens.

The next line contains n integers a_1, a_2, \dots, a_n ($0 \leq a_i \leq 10^9$) — the initial balances of citizens.

The next line contains a single integer q ($1 \leq q \leq 2 \cdot 10^5$) — the number of events.

Each of the next q lines contains a single event. The events are given in chronological order.

Each event is described as either $1 \ p \ x$ ($1 \leq p \leq n, 0 \leq x \leq 10^9$), or $2 \ x$ ($0 \leq x \leq 10^9$). In the first case we have a receipt that the balance of the p -th person becomes equal to x . In the second case we have a payoff with parameter x .

Output

Print n integers — the balances of all citizens after all events.

Examples

input	Copy
4 1 2 3 4 3 2 3 1 2 2 2 1	
output	Copy
3 2 3 4	

input	Copy
5 3 50 2 1 10 3 1 2 0 2 8 1 3 20	
output	Copy
8 8 20 8 10	

Note

In the first example the balances change as follows: $1 \ 2 \ 3 \ 4 \rightarrow 3 \ 3 \ 3 \ 4 \rightarrow 3 \ 2 \ 3 \ 4 \rightarrow 3 \ 2 \ 3 \ 4$

In the second example the balances change as follows: $3 \ 50 \ 2 \ 1 \ 10 \rightarrow 3 \ 0 \ 2 \ 1 \ 10 \rightarrow 8 \ 8 \ 8 \ 8 \ 10 \rightarrow 8 \ 8 \ 20 \ 8 \ 10$

Codeforces Round #576 (Div. 2)

Finished

Practice



→ Virtual participation

Virtual contest is a way to take part in past contest, as close as possible to participation on time. It is supported only ACM-ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you - solve these problems in the archive. If you just want to solve some problem from a contest, a virtual contest is not for you - solve this problem in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Start virtual contest

→ Practice

You are registered for practice. You can solve problems unofficially. Results can be found in the contest status and in the bottom of standings.

→ Clone Contest to Mashup

You can clone this contest to a mashup.

Clone Contest

→ Submit?

Language: GNU G++11 5.1.0

Choose file: 浏览...

Be careful: there is 50 points penalty for submission which fails the pretests or resubmission (except failure on the first test, denial of judgement or similar verdicts). "Passed pretests" submission verdict doesn't guarantee that the solution is absolutely correct and it will pass system tests.

Submit

→ Last submissions



Submission	Time	Verdict
58085540	Jul/31/2019 18:26	Accepted

→ Problem tags

data structures implementation

No tag edit access

→ Contest materials

- Announcement (en) 
- Tutorial (en) 

[Codeforces](#) (c) Copyright 2010-2019 Mike Mirzayanov
The only programming contests Web 2.0 platform
Server time: Aug/01/2019 13:02:53^{UTC+8} (g1).
Desktop version, switch to [mobile version](#).
[Privacy Policy](#)

Supported by



ITMO UNIVERSITY