Wild operations (wild)

Filippo wants to test Francesco's ability in handling wild array operations, therefore he has given him an array $A_0, ..., A_{N-1}$ of length N.

Now Filippo will ask Francesco to perform some operations on the array, where each operation can be either:

- *change* the value of A_p to x, for some integer x and valid index p.
- *perturb* the range $[l, r]$, i.e. to set $A_p = \max(A_p, A_{p-1})$ **simultaneously** for all $l < p \le r$.

At any moment Filippo may ask Francesco to tell him the value of A_p for some valid index p.

Francesco is very busy, so he decided to ask for your help to answer Filippo's questions.

Implementation

You must submit a single file with extension `.cpp`.

⇦ Among the attachments for this task you will find a template `wild.cpp` with an example implementation.

You must implement the following functions:

C++	`void init(int N, vector<int> A);`

- This function is called once, at the beginning of your program execution.
- The integer N is the length of the array.
- The array A, indexed from 0 to $N - 1$, is the initial array Filippo has chosen.

C++	`void change(int p, int x);`

- This function is called many times during the execution of your program, when Filippo performs a change.
- The integer p is the index of the changed value in the array.
- The integer x is the new value to be assigned.

C++	`void perturb(int l, int r);`

- This function is called many times during the execution of your program, when Filippo perturbs a range.
- The integer l is the left end of the range perturbed by Filippo.
- The integer r is the right end of the range perturbed by Filippo.

C++	`int calc(int p);`

- This function is called many times during the execution of your program, when Filippo asks the value of an element of the array.
- The integer p is the index of the element asked by Filippo.
- The function should return the value of A_p after applying all previous operations.

Sample Grader

A simplified version of the grader used during the correction is available in the directory related to this problem. You can use it to test your solutions locally. The sample grader reads input data from `stdin`, calls the function you need to implement, and writes to `stdout` in the following format.

Let Q be the total number of changes, perturbations and questions done by Filippo.
Then, the input file is made up of $2 + Q$ lines, containing:

- Line 1: the integers N, Q.
- Line 2: N integers $A_0, ..., A_{N-1}$, the initial values of the array.
- Line $3 + i$ $(0 \leq i < Q)$: 2 or 3 integers, in one of the following formats:
  - $1\ p\ x$: meaning Filippo changes A_p to x.
  - $2\ l\ r$: meaning Filippo perturbs the range $[l, r]$;
  - $3\ p$: meaning Filippo asks the value of A_p.

The output file is made of Q_3 lines (where Q_3 is the number of calls to `calc`) containing the values returned by the function `calc`.

Constraints

- $1 \leq N \leq 400\,000$.
- $0 \leq Q \leq 400\,000$.
- $1 \leq A_i \leq 10^9$ for all $0 \leq i < N$.
- $0 \leq p < N$ in every call to `change` and `calc`.
- $0 \leq l < r \leq N - 1$ in every call to `perturb`.
- $1 \leq x \leq 10^9$ in every call to `change`.

Scoring

Your program will be tested on several test cases grouped into subtasks. To obtain the points for a subtask, you must solve all test cases in it correctly.

Let Q_1 be the number of calls to the function `change` in a testcase, then:

- **Subtask 0** [**0 points**]: Example.
- **Subtask 1** [**15 points**]: The function `change` is never called; $l = 0$, $r = N - 1$ in every call to `perturb`.
- **Subtask 2** [**16 points**]: $A_i \leq 10$ for all $0 \leq i < N$ and $x \leq 10$ for all calls to `change`.
- **Subtask 3** [**13 points**]: Calls to the function `change` do not decrease the values $(x \geq A_p)$, $Q_1 \leq 1000$ and $l = 0$, $r = N - 1$ in every call to `perturb`.
- **Subtask 4** [**22 points**]: The function `change` is never called.
- **Subtask 5** [**14 points**]: Calls to the function `change` do not decrease the values $(x \geq A_p)$, $Q_1 \leq 1000$.
- **Subtask 6** [**20 points**]: No additional constraints.

Examples

stdin	stdout
10 28	1
5 1 7 8 3 2 5 6 9 4	3
1 1 1	1
1 0 1	7
2 0 1	8
2 2 6	1
1 6 5	8
2 2 9	3
2 2 5	6
2 4 5	4
1 4 5	9
2 3 8	
1 8 4	
3 0	
1 6 3	
1 4 1	
2 5 7	
1 0 3	
2 4 5	
1 6 3	
3 0	
3 1	
3 2	
3 3	
3 4	
3 5	
3 6	
3 7	
3 8	
3 9	

Explanation

We start with the array $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Event 1: Filippo changes A_1 to 1 (it was already 1): the new array is $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Event 2: Filippo changes A_0 to 1: the new array is $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Event 3: Filippo perturbs $[0, 1]$: the new array is $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Event 4: Filippo perturbs $[2, 6]$: the new array is $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

From event 19 onward, Filippo only asks for values in the array without performing changes or perturbations. At this point the array is $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.