Wild operations (wild)

Филиппо хочет проверить, насколько хорошо Франческо справляется с безумными операциями над массивами, и поэтому дал ему массив $A_0, ..., A_{N-1}$ длины N.

Теперь Филиппо будет просить Франческо выполнять некоторые операции с массивом. Каждая операция может быть одной из следующих:

- *change* (изменение): поменять значение A_p на x, для некоторого целого числа x и корректного индекса p.
- *perturb* (возмущение): для отрезка $[l, r]$, то есть, присвоить $A_p = \max(A_p, A_{p-1})$ **одновременно** для всех p от $l + 1$ до r включительно.

В любой момент Филиппо может спросить у Франческо значение элемента A_p для некоторого корректного индекса p.

Франческо очень занят, поэтому он решил попросить вас помочь ему отвечать на вопросы Филиппо.

Implementation

Вы должны отправить один файл с расширением `.cpp`.

> ⇦ Среди приложенных к задаче файлов вы найдете шаблон `wild.cpp` с примером реализации.

Вам нужно реализовать следующие функции:

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

- Эта функция вызывается один раз, в самом начале выполнения вашей программы.
- Целое число N — это длина массива.
- Массив A, индексируемый от 0 до $N - 1$, — это начальный массив, который выбрал Филиппо.

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

- Эта функция вызывается много раз во время выполнения вашей программы, когда Филиппо делает изменение.
- Целое число p — это индекс изменяемого элемента в массиве.
- Целое число x — это новое значение, которое нужно присвоить.

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

- Эта функция вызывается много раз во время выполнения вашей программы, когда Филиппо применяет операцию “perturb”.
- Целое число l — это левая граница отрезка, к которому применяется операция.
- Целое число r — это правая граница отрезка, к которому применяется операция.

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

- Эта функция вызывается много раз во время выполнения вашей программы, когда Филиппо спрашивает значение элемента массива.
- Целое число p — это индекс элемента, о котором спрашивает Филиппо.
- Функция должна вернуть значение A_p после применения всех предыдущих операций.

Пример грейдера

Упрощенная версия грейдера, который будет использоваться при проверке, доступна в директории с задачей. Вы можете использовать её для локального тестирования своих решений. Пример грейдера считывает входные данные из `stdin`, вызывает функции, которые вам нужно реализовать, и записывает результат в `stdout` в следующем формате.

Пусть Q — общее количество изменений, возмущений и запросов, сделанных Филиппо.
Тогда входной файл состоит из $2 + Q$ строк, содержащих:

- Строка 1: целые числа N, Q.
- Строка 2: N целых чисел $A_0, ..., A_{N-1}$ — начальные значения массива.
- Строка $3 + i$ $(0 \le i < Q)$: 2 или 3 целых числа в одном из следующих форматов:
  - $1\ p\ x$: означает, что Филиппо меняет A_p на x.
  - $2\ l\ r$: означает, что Филиппо применяет "perturb" к отрезку $[l, r]$.
  - $3\ p$: означает, что Филиппо спрашивает значение A_p.

Выходной файл состоит из Q_3 строк (где Q_3 — количество вызовов `calc`), содержащих значения, возвращенные функцией `calc`.

Constraints

- $1 \le N \le 400\,000$.
- $0 \le Q \le 400\,000$.
- $1 \le A_i \le 10^9$ для всех $0 \le i < N$.
- $0 \le p < N$ в каждом вызове `change` и `calc`.
- $0 \le l < r \le N - 1$ в каждом вызове `perturb`.
- $1 \le x \le 10^9$ в каждом вызове `change`.

Scoring

Ваша программа будет протестирована на нескольких наборах тестов, сгруппированных в подзадачи. Чтобы получить баллы за подзадачу, вы должны правильно решить все тесты в ней.

Пусть Q_1 — это количество вызовов функции `change` в одном тесте, тогда:

- **Subtask 0 [0 points]**: Пример.
- **Subtask 1 [15 points]**: Функция `change` никогда не вызывается; $l = 0$, $r = N - 1$ в каждом вызове `perturb`.
- **Subtask 2 [16 points]**: $A_i \le 10$ для всех $0 \le i < N$ и $x \le 10$ для всех вызовов `change`.
- **Subtask 3 [13 points]**: Вызовы функции `change` не уменьшают значения $(x \ge A_p)$, $Q_1 \le 1000$ и $l = 0$, $r = N - 1$ в каждом вызове `perturb`.
- **Subtask 4 [22 points]**: Функция `change` никогда не вызывается.
- **Subtask 5 [14 points]**: Вызовы функции `change` не уменьшают значения $(x \ge A_p)$, $Q_1 \le 1000$.
- **Subtask 6 [20 points]**: Нет дополнительных ограничений.

Examples

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

Explanation

Мы начинаем с массива $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Событие 1: Филиппо меняет A_1 на 1 (он уже был равен 1): новый массив — $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Событие 2: Филиппо меняет A_0 на 1: новый массив — $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Событие 3: Филиппо применяет "perturb" к $[0, 1]$: новый массив — $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Событие 4: Филиппо применяет "perturb" к $[2, 6]$: новый массив — $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

Начиная с события 19, Филиппо только запрашивает значения в массиве, не выполняя изменений или "perturb"-операций. На этот момент массив выглядит так: $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.