Wild operations (wild)

Филип иска да провери способността на Франческо да обработва диви операции върху масиви. Затова той му дава масива $A_0, ..., A_{N-1}$ с дължина N.

Сега Филип ще поиска от Франческо да извърши операции върху масива, където всяка операция е някоя от следните:

- *change* - стойността на A_p се променя на x, за някое цяло число x и валиден индекс p.
- *perturb* - пертурбация на интервала $[l, r]$, т.е. $A_p = \max(A_p, A_{p-1})$ **едновременно** за всяко $l < p \leq r$.

По всяко време Филип може да попита Франческо за стойността на A_p за някой валиден индекс p.

Франческо е много зает, така че той решава да попита Вас за помощ при отговарянето на въпросите на Филип.

Имплементация

Трябва да изпратите един файл с разширение `.cpp`.

> ⇦ Измежду прикачените файлове за тази задача ще намерите шаблон `wild.cpp` с примерна имплементация.

Трябва да напишете следните функции:

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

- Тази функция се извиква веднъж в началото на изпълнението на програмата.
- Цялото число N е дължината на масива.
- Векторът A, индексиран от 0 до $N-1$, е началният масив, избран от Филип.

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

- Тази функция се извиква неколкоратно по време на изпълнението на програмата, когато Филип извършва *change*.
- Цялото число p е индексът на елемента в масива, за който се извършва операцията.
- Цялото число x е стойността, която се присвоява.

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

- Тази функция се извиква неколкократно по време на изпълнението на програмата, когато Филип извършва *perturb* на интервал.
- Цялото число l е левият край на интервала, за който се извършва операцията.
- Цялото число r е десният край на интервала, за който се извършва операцията.

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

- Тази функция се извиква неколкократно по време на изпълнението на програмата, когато Филип пита за стойността на елемент на масива.

- Цялото число p е индексът на елемента, за който пита Филип.
- Функцията трябва да върне стойността на A_p след извършване на всички предишни операции.

Примерен грейдър

Опростена версия на грейдъра, използван при оценяването, е налична в директорията на задачата. Можете да го използвате, за да тествате решенията си локално. Примерният грейдър чете входните данни от `stdin`, извиква функциите, които трябва да имплементирате, и отпечатва резултатите на `stdout` в следния формат.

Нека Q е общият брой на `change`, `perturb` и въпросите, зададени от Филип.
Тогава входният файл трябва да е с $2 + Q$ реда, по-точно:

- Ред 1: целите числа N, Q.
- Ред 2: N цели числа $A_0, ..., A_{N-1}$, началните стойности на масива.
- Ред $3 + i$ $(0 \leq i < Q)$: 2 или 3 цели числа, в някой от следните формати:
  - 1 p x: задаващ операцията *change* за промяна на A_p на x.
  - 2 l r: задаващ операцията *perturb* за пертурбация на интервала $[l, r]$;
  - 3 p: задаващ въпрос на Филип за стойността на A_p.

Изходният файл има Q_3 реда (където Q_3 е броят на извикванията на `calc`), които съдържат стойностите, върнати от извикванията на `calc`.

Ограничения

- $1 \leq N \leq 400\,000$.
- $0 \leq Q \leq 400\,000$.
- $1 \leq A_i \leq 10^9$ за всяко $0 \leq i < N$.
- $0 \leq p < N$ за всяко извикване на `change` и `calc`.
- $0 \leq l < r \leq N - 1$ за всяко извикне на `perturb`.
- $1 \leq x \leq 10^9$ за всяко извикване на `change`.

Оценяване

Вашата програма ще бъде оценена на няколко теста, групирани в подзадачи. За да получите точките за дадена подзадача, трябва да решите вярно всички тестове в нея.

Нека Q_1 е броят на извикванията на функция `change` за тест. Тогава:

- **Подзадача 0 [0 точки]**: Примерът.
- **Подзадача 1 [15 точки]**: Функцията `change` не се извиква; $l = 0$, $r = N - 1$ за всяко извикване на `perturb`.
- **Подзадача 2 [16 точки]**: $A_i \leq 10$ за всяко $0 \leq i < N$ и $x \leq 10$ за всички извиквания на `change`.
- **Подзадача 3 [13 точки]**: Извикванията на функцията `change` не намаляват стойностите $(x \geq A_p)$, $Q_1 \leq 1000$ и $l = 0$, $r = N - 1$ за всяко извикване на `perturb`.
- **Подзадача 4 [22 точки]**: Функцията `change` не се извиква.
- **Подзадача 5 [14 точки]**: Извикванията на функция `change` не намаляват стйностите $(x \geq A_p)$, $Q_1 \leq 1000$.
- **Подзадача 6 [20 точки]**: Няма допълнителни ограничения.

Примерни входове/изходи

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

Обяснение

Започваме с масива $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Събитие 1: Филип променя A_1 на 1 (преди това също е 1): новият масив е $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Събитие 2: Филип променя A_0 на 1: новият масив е $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Събитие 3: Филип извършва *perturb* на интервала $[0, 1]$: новият масив е $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Събитие 4: Филип извършва *perturb* на интервала $[2, 6]$: новият масив е $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

От събитие 19 нататък, Филип единствено пита за стойностите на масива, без да извърша някоя от операциите. Масивът по това време е $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.