

## Wild operations (wild)

Filippo vrea să-i testeze lui Francesco abilitatea de a se descurca cu operații pe șiruri sălbatice, așa că i-a dat un șir  $A_0, \dots, A_{N-1}$  de lungime  $N$ .

Acum Filippo îi va cere lui Francesco să efectueze câteva operații pe șir, unde fiecare operație poate fi:

- *schimbă* valoarea lui  $A_p$  la  $x$ , pentru un întreg  $x$  și un index valid  $p$ .
- *perturbează* intervalul  $[l, r]$ , adică să seteze  $A_p = \max(A_p, A_{p-1})$  **simultan** pentru toți  $l < p \leq r$ .

În orice moment, Filippo îi poate cere lui Francesco să-i spună valoarea lui  $A_p$  pentru un index valid  $p$ .

Francesco e foarte ocupat, așa că a decis să-ți ceară ajutorul pentru a răspunde întrebărilor lui Filippo.

## Implementare

Trebuie să trimiți un singur fișier cu extensia `.cpp`.



Printre atașamentele pentru această problemă vei găsi un șablon `wild.cpp` cu o implementare exemplu.

Trebuie să implementezi următoarele funcții:

C++

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

- Această funcție este apelată o singură dată, la începutul execuției programului tău.
- Întregul  $N$  este lungimea șirului.
- Șirul  $A$ , indexat de la 0 la  $N - 1$ , este șirul inițial ales de Filippo.

C++

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

- Această funcție este apelată de multe ori pe parcursul execuției programului tău, când Filippo efectuează o modificare.
- Întregul  $p$  este indexul valorii modificate în șir.
- Întregul  $x$  este noua valoare de atribuit.

C++

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

- Această funcție este apelată de multe ori pe parcursul execuției programului tău, când Filippo perturbează un interval.
- Întregul  $l$  este capătul stâng al intervalului perturbat de Filippo.
- Întregul  $r$  este capătul drept al intervalului perturbat de Filippo.

C++

```
int calc(int p);
```

- Această funcție este apelată de multe ori pe parcursul execuției programului tău, când Filippo cere valoarea unui element din șir.
- Întregul  $p$  este indexul elementului cerut de Filippo.
- Funcția ar trebui să returneze valoarea lui  $A_p$  după aplicarea tuturor operațiilor anterioare.

## Evaluator exemplu

O versiune simplificată a evaluatorului folosit la corectare este disponibilă în directorul aferent acestei probleme. O poți folosi pentru a-ți testa soluțiile local. Evaluatorul exemplu citește datele de intrare din `stdin`, apelează funcția pe care trebuie să o implementezi și scrie în `stdout` în următorul format.

Fie  $Q$  numărul total de modificări, perturbații și întrebări făcute de Filippo.

Atunci, fișierul de intrare este compus din  $2 + Q$  linii, conținând:

- Linia 1: întregii  $N, Q$ .
- Linia 2:  $N$  întregi  $A_0, \dots, A_{N-1}$ , valorile inițiale ale șirului.
- Linia  $3 + i$  ( $0 \leq i < Q$ ): 2 sau 3 întregi, într-unul din următoarele formate:
  - ▶ 1  $p$   $x$ : înseamnă că Filippo schimbă  $A_p$  la  $x$ .
  - ▶ 2  $l$   $r$ : înseamnă că Filippo perturbază intervalul  $[l, r]$ ;
  - ▶ 3  $p$ : înseamnă că Filippo cere valoarea lui  $A_p$ .

Fișierul de ieșire este compus din  $Q_3$  linii (unde  $Q_3$  este numărul de apeluri către `calc`) conținând valorile returnate de funcția `calc`.

## Constrângeri

- $1 \leq N \leq 400\,000$ .
- $0 \leq Q \leq 400\,000$ .
- $1 \leq A_i \leq 10^9$  pentru toți  $0 \leq i < N$ .
- $0 \leq p < N$  în fiecare apel la `change` și `calc`.
- $0 \leq l < r \leq N - 1$  în fiecare apel la `perturb`.
- $1 \leq x \leq 10^9$  în fiecare apel la `change`.

## Punctaj

Programul tău va fi testat pe mai multe cazuri de test grupate în subtaskuri. Pentru a obține punctele pentru un subtask, trebuie să rezolvi corect toate cazurile de test din el.

Fie  $Q_1$  numărul de apeluri către funcția `change` într-un caz de test, atunci:

- **Subtask-ul 0 [ 0 puncte]**: Exemplu.
- **Subtask-ul 1 [15 puncte]**: Funcția `change` nu este apelată niciodată;  $l = 0, r = N - 1$  în fiecare apel la `perturb`.
- **Subtask-ul 2 [16 puncte]**:  $A_i \leq 10$  pentru toți  $0 \leq i < N$  și  $x \leq 10$  pentru toate apelurile la `change`.
- **Subtask-ul 3 [13 puncte]**: Apelurile la funcția `change` nu scad valorile ( $x \geq A_p$ ),  $Q_1 \leq 1000$  și  $l = 0, r = N - 1$  în fiecare apel la `perturb`.
- **Subtask-ul 4 [22 puncte]**: Funcția `change` nu este apelată niciodată.
- **Subtask-ul 5 [14 puncte]**: Apelurile la funcția `change` nu scad valorile ( $x \geq A_p$ ),  $Q_1 \leq 1000$ .
- **Subtask-ul 6 [20 puncte]**: Fără constrângeri suplimentare.

## Exemple de intrare/ieșire

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	

## Explicație

Începem cu șirul  $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$ .

- Evenimentul 1: Filippo schimbă  $A_1$  la 1 (era deja 1): noul șir este  $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$ .
- Evenimentul 2: Filippo schimbă  $A_0$  la 1: noul șir este  $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$ .
- Evenimentul 3: Filippo perturbază  $[0, 1]$ : noul șir este  $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$ .
- Evenimentul 4: Filippo perturbază  $[2, 6]$ : noul șir este  $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$ .

De la evenimentul 19 încolo, Filippo cere doar valori din șir fără a efectua modificări sau perturbații. În acest punct, șirul este  $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$ .