Wild operations (wild)

Filippo želi preizkusiti Francescovo sposobnost upravljanja z divjimi operacijami nad polji, zato mu da polje $A_0, ..., A_{N-1}$ dolžine N.

Zdaj bo Filippo Francescu naročil izvajanje operacij nad polji, kjer lahko vsaka operacija:

- *spremeni* vrednost A_p v x, kjer je x celo število in p veljaven indeks.
- *perturbiraj* območje $[l, r]$, tj. nastavi $A_p = \max(A_p, A_{p-1})$ **istočasno** za vse $l < p \leq r$. (Najprej se izračunajo vsi maksimumi, nato se pa nastavijo elementi)

Filippo lahko kadarkoli vpraša Francesca po vrednosti poljubnega A_p.

Francesco je zelo zaposlen, zato te prosi za pomoč pri odgovarjanju na Filippova vprašanja.

Implementacija

Oddati morate eno datoteko s končnico `.cpp`.

⇦ Med prilogami te naloge boste našli predlogo `wild.cpp` z zgledom implementacije.

Implementirati morate naslednje funkcije:

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

- Ta procedura se kliče enkrat, na začetku izvajanja programa.
- N je dolžina polja.
- Polje A, indeksirano od 0 do $N - 1$, je začetno polje, ki ga izbere Filippo.

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

- Ta procedura se med izvajanjem vašega programa kliče večkrat, vsakič ko Filippo izvede spremembo.
- p je indeks v polju, na katerem se spremenjena vrednost.
- x je nova vrednost, ki naj se dodeli.

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

- Ta procedura se med izvajanjem vašega programa kliče večkrat, vsakič ko Filippo perturbira območje.
- l je levi rob območja, katerega Filippo perturbira.
- r je desni rob območja, katerega Filippo perturbira.

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

- Ta funkcija se med izvajanjem vašega programa kliče večkrat, vsakič ko Filippo vpraša za vrednost nekega elementa polja.
- p je indeks elementa, ki zanima Filippa.
- Funkcija naj vrne vrednost A_p, upoštevajoč vse prejšnje operacije.

Vzorčni ocenjevalnik

Poenostavljena različica ocenjevalnika, ki se uporablja med ocenjevanjem, je na voljo v imeniku, povezanem s to nalogo. Uporabite ga lahko za lokalno testiranje vaših rešitev. Vzorčni ocenjevalnik prebere vhodne podatke iz `stdin`, kliče funkcijo, ki jo morate implementirati, ter piše v `stdout` v naslednji obliki.

Naj bo Q skupno število sprememb, perturbacij in vprašanj, ki jih izvede Filippo.
Vhodna datoteka je sestavljena iz vrstic $2 + Q$, ki vsebujejo:

- Vrstica 1: cela števila N, Q.
- Vrstica 2: N celih števil $A_0, ..., A_{N-1}$, začetne vrednosti polja.
- Vrstice $3 + i$ ($0 \leq i < Q$): 2 ali 3 celih števil, v eni od naslednjih oblik:
  - 1 p x: pomeni, da Filippo spremeni A_p na x.
  - 2 l r: pomeni, da Filippo perturbira območje $[l, r]$;
  - 3 p: pomeni, da Filippo vpraša za vrednost A_p.

Izhodna datoteka je sestavljena iz vrstic Q_3 (kjer je Q_3 število klicev `calc`) in vsebuje vrednosti, ki jih vrne funkcija `calc`.

Omejitve

- $1 \leq N \leq 400\,000$.
- $0 \leq Q \leq 400\,000$.
- $1 \leq A_i \leq 10^9$ za vse $0 \leq i < N$.
- $0 \leq p < N$ pri vsakem klicu `change` in `calc`.
- $0 \leq l < r \leq N - 1$ pri vsakem klicu `perturb`.
- $1 \leq x \leq 10^9$ pri vsakem klicu `change`.

Točkovanje

Vaš program bo testiran na več testnih primerih, združenih v podnaloge. Za pridobitev točk za podnalogo morate v njej pravilno rešiti vse testne primere.

Naj bo Q_1 število klicev funkcije `change` v testnem primeru, potem:

- **Podnaloga 0 [0 točk]**: Primer.
- **Podnaloga 1 [15 točk]**: Funkcija `change` se nikoli ne kliče; $l = 0$, $r = N - 1$ pri vsakem klicu `perturb`.
- **Podnaloga 2 [16 točk]**: $A_i \leq 10$ za vse $0 \leq i < N$ in $x \leq 10$ pri vseh klicih `change`.
- **Podnaloga 3 [13 točk]**: Klici funkcije `change` ne zmanjšujejo vrednosti ($x \geq A_p$), $Q_1 \leq 1000$ in $l = 0$, $r = N - 1$ pri vsakem klicu `perturb`.
- **Podnaloga 4 [22 točk]**: Funkcija `change` se nikoli ne kliče.
- **Podnaloga 5 [14 točk]**: Klici funkcije `change` ne zmanjšujejo vrednosti ($x \geq A_p$), $Q_1 \leq 1000$.
- **Podnaloga 6 [20 točk]**: Brez dodatnih omejitev.

Primeri vhoda/izhoda

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

Razlaga

Začnemo s poljem $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Dogodek 1: Filippo spremeni A_1 na 1 (je že bil 1): novo polje je $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Dogodek 2: Filippo spremeni A_0 na 1: novo polje je $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Dogodek 3: Filippo perturbira $[0, 1]$: novo polje je $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Dogodek 4: Filippo perturbira $[2, 6]$: novo polje je $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

Od dogodka 19 naprej Filippo samo sprašuje po vrednostih v polju, brez sprememb ali perturbacij. V tem trenutku je polje $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.