Wild operations (wild)

Souris Binna veut tester la capacité de Souris Stofl à gérer des opérations bizarres sur des tableaux. Pour ça, elle lui a donné un tableau $A_0, ..., A_{N-1}$ de longueur N.

Maintenant, Binna va demander à Stofl d'effectuer quelques opérations sur le tableau. Chaque opération peut être soit :

- *changer* la valeur de A_p en x, pour un certain entier x et un indice valide p.
- *perturber* l'intervalle $[l, r]$, c'est-à-dire de mettre $A_p = \max(A_p, A_{p-1})$ **simultanément** pour tout $l < p \leq r$.

À tout moment, Binna peut demander à Stofl de lui donner la valeur de A_p pour un certain indice valide p.

Stofl est très occupé, alors il a décidé de te demander de l'aide pour répondre aux questions de Binna.

Implémentation

Tu dois soumettre un unique fichier avec l'extension `.cpp`.

> ⇦ Parmi les pièces jointes de ce problème, tu trouveras un template `wild.cpp` avec un exemple d'implémentation.

Tu dois implémenter les fonctions suivantes :

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

- Cette fonction est appelée une seule fois, au début de l'exécution de ton programme.
- L'entier N est la longueur du tableau.
- Le tableau A, indexé de 0 à $N-1$, est le tableau initial que Binna a choisi.

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

- Cette fonction est appelée plusieurs fois pendant l'exécution de ton programme, lorsque Binna effectue un changement.
- L'entier p est l'indice de la valeur modifiée dans le tableau.
- L'entier x est la nouvelle valeur à assigner.

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

- Cette fonction est appelée plusieurs fois pendant l'exécution de ton programme, lorsque Binna perturbe un intervalle.
- L'entier l est l'extrémité gauche de l'intervalle perturbé par Binna.
- L'entier r est l'extrémité droite de l'intervalle perturbé par Binna.

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

- Cette fonction est appelée plusieurs fois pendant l'exécution de ton programme, lorsque Binna demande la valeur d'un élément du tableau.

- L'entier p est l'indice de l'élément demandé par Binna.
- La fonction doit retourner la valeur de A_p après avoir appliqué toutes les opérations précédentes.

Grader d'Exemple

Une version simplifiée du grader utilisé lors de la correction est disponible dans le dossier relatif à ce problème. Tu peux l'utiliser pour tester tes solutions localement. Le grader d'exemple lit les données d'entrée depuis `stdin`, appelle les fonctions que tu dois implémenter, et écrit la sortie dans `stdout` au format suivant.

Soit Q le nombre total de changements, perturbations et questions posées par Binna.
Le fichier d'entrée est alors composé de $2 + Q$ lignes, contenant :

- Ligne 1 : les entiers N, Q.
- Ligne 2 : N entiers $A_0, ..., A_{N-1}$, les valeurs initiales du tableau.
- Ligne $3 + i$ $(0 \leq i < Q)$: 2 ou 3 entiers, dans l'un des formats suivants :
  - 1 p x : signifie que Binna change A_p en x.
  - 2 l r : signifie que Binna perturbe l'intervalle $[l, r]$;
  - 3 p : signifie que Binna demande la valeur de A_p.

Le fichier de sortie est composé de Q_3 lignes (où Q_3 est le nombre d'appels à `calc`) contenant les valeurs retournées par la fonction `calc`.

Contraintes

- $1 \leq N \leq 400\,000$.
- $0 \leq Q \leq 400\,000$.
- $1 \leq A_i \leq 10^9$ pour tout $0 \leq i < N$.
- $0 \leq p < N$ dans chaque appel à `change` et `calc`.
- $0 \leq l < r \leq N - 1$ dans chaque appel à `perturb`.
- $1 \leq x \leq 10^9$ dans chaque appel à `change`.

Score

Ton programme sera testé sur plusieurs cas de test regroupés en sous-problèmes. Pour obtenir les points d'un sous-problème, tu dois résoudre correctement tous les cas de test qu'il contient.

Soit Q_1 le nombre d'appels à la fonction `change` dans un cas de test, alors :

- **Sous-tâche 0 [0 points]**: Exemple.
- **Sous-tâche 1 [15 points]**: La fonction `change` n'est jamais appelée ; $l = 0$, $r = N - 1$ dans chaque appel à `perturb`.
- **Sous-tâche 2 [16 points]**: $A_i \leq 10$ pour tout $0 \leq i < N$ et $x \leq 10$ pour tous les appels à `change`.
- **Sous-tâche 3 [13 points]**: Les appels à la fonction `change` n'abaissent pas les valeurs $(x \geq A_p)$, $Q_1 \leq 1000$ et $l = 0$, $r = N - 1$ dans chaque appel à `perturb`.
- **Sous-tâche 4 [22 points]**: La fonction `change` n'est jamais appelée.
- **Sous-tâche 5 [14 points]**: Les appels à la fonction `change` n'abaissent pas les valeurs $(x \geq A_p)$, $Q_1 \leq 1000$.
- **Sous-tâche 6 [20 points]**: Aucune contrainte additionnelle.

Exemples

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

Explication

On commence avec le tableau $A = [5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.

- Événement 1 : Binna change A_1 en 1 (il valait déjà 1) : le nouveau tableau est $[5, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Événement 2 : Binna change A_0 en 1 : le nouveau tableau est $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Événement 3 : Binna perturbe $[0, 1]$: le nouveau tableau est $[1, 1, 7, 8, 3, 2, 5, 6, 9, 4]$.
- Événement 4 : Binna perturbe $[2, 6]$: le nouveau tableau est $[1, 1, 7, 8, 8, 3, 5, 6, 9, 4]$.

À partir de l'événement 19, Binna ne fait que demander des valeurs dans le tableau sans effectuer de changements ou de perturbations. À ce stade, le tableau est $[3, 1, 7, 8, 1, 8, 3, 6, 4, 9]$.