



## Wall

Jian-Jia baut eine Mauer, indem er Ziegelsteine derselben Größe zusammensetzt. Diese Mauer besteht aus  $n$  Säulen von Ziegelsteinen, die von links nach rechts von 0 bis  $n - 1$  nummeriert sind. Die Säulen können unterschiedliche Höhen haben. Die Höhe einer Säule ist bestimmt durch die Anzahl der Ziegelsteine aus denen sie gebaut ist.

Jian-Jia baut die Mauer wie folgt. Am Anfang sind die Säulen leer (sie bestehen aus 0 Ziegelsteinen). Danach folgt Jian-Jia einem Bauprozess, der aus  $k$  Phasen besteht. In jeder Phase wird Jian-Jia entweder Ziegelsteine *hinzufügen* oder *entfernen*. Der Bauprozess ist abgeschlossen, wenn die  $k$  Phasen abgeschlossen sind. In jeder Phase erhält Jian-Jia folgende Angaben: Einen zusammenhängenden Bereich von benachbarten Säulen und die gewünschte Höhe  $h$ . Aufgrund dieser Angaben handelt er wie folgt:

- In einer *Hinzufügephase* fügt Jian-Jia Ziegelsteine zu jenen Säulen in dem gegebenen Bereich hinzu, deren Höhe kleiner als  $h$  ist, bis deren Höhe exakt  $h$  erreicht hat. Die Säulen, deren Höhe mindestens  $h$  ist, verändert er nicht.
- In einer *Entfernphase* entfernt Jian-Jia Ziegelsteine von jenen Säulen in dem gegebenen Bereich, deren Höhe größer als  $h$  ist, bis deren Höhe exakt  $h$  erreicht hat. Die Säulen, deren Höhe höchstens  $h$  ist, verändert er nicht.

Deine Aufgabe ist es, die endgültige Form der Mauer zu bestimmen.

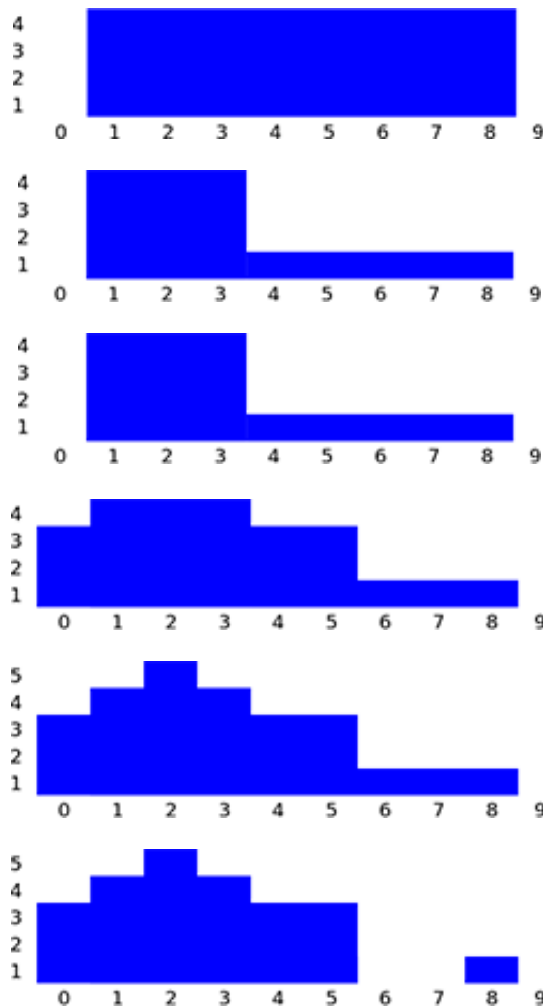
## Beispiel

Wir nehmen an, es gibt 10 Säulen und 6 Bauphasen. Alle Bereichsangaben enthalten die Grenzen. Die untenstehenden Diagramme zeigen die jeweilige Form der Mauer nach jeder Phase.

Phase	Typ	Bereich	Höhe
0	hinzufügen	Säulen 1 bis 8	4
1	entfernen	Säulen 4 bis 9	1
2	entfernen	Säulen 3 bis 6	5
3	hinzufügen	Säulen 0 bis 5	3
4	hinzufügen	Säule 2	5
5	entfernen	Säulen 6 bis 7	0

Da alle Säulen am Anfang leer sind, werden nach der Phase 0 die Säulen 1 bis 8 aus jeweils 4 Ziegelsteinen bestehen. Säulen 0 und 9 bleiben leer. In Phase 1 werden Ziegelsteine von den Säulen 4 bis 8 solange entfernt bis diese Säulen nur mehr aus einem Ziegelstein bestehen und Säule 9 bleibt leer. Die Säulen 0 bis 3, die ausserhalb des angegebenen Bereichs liegen, bleiben unverändert. Die Phase 2 hinterlässt keine Veränderungen, da keine der Säulen 3 bis 6 mehr als 5 Ziegelsteine enthält. In Phase 3 erhöht sich die Anzahl der Ziegelsteine in den Säulen 0, 4 und 5 auf jeweils 3. Nach Phase

4 befinden sich 5 Ziegelsteine in Säule 2. In Phase 5 werden alle Ziegelsteine aus den Säulen 6 und 7 entfernt.



## Aufgabe

Gegeben ist die Beschreibung von  $k$  Phasen. Berechne die Anzahl der Ziegelsteine für jede Säule nachdem alle Phasen abgeschlossen sind. Erstelle dazu die Funktion `buildWall`.

- `buildWall(n, k, op, left, right, height, finalHeight)`
  - $n$ : die Anzahl der Säulen in der Mauer.
  - $k$ : die Anzahl der Phasen.
  - `op`: Array der Länge  $k$ ; `op[i]` ist der Typ der Phase  $i$ : 1 für hinzufügen und 2 für entfernen, für  $0 \leq i \leq k - 1$ .
  - `left` und `right`: Arrays der Länge  $k$ ; Der Bereich der Säulen für die Phase  $i$  beginnt mit Säule `left[i]` und endet mit Säule `right[i]` (inklusive der beiden Grenzen `left[i]` und `right[i]`), für  $0 \leq i \leq k - 1$ . Es gilt immer `left[i] ≤ right[i]`.
  - `height`: Array der Länge  $k$ ; `height[i]` ist die gewünschte Höhe der Phase  $i$ , für  $0 \leq i \leq k - 1$ .
  - `finalHeight`: Array der Länge  $n$ ; hier sollst du die Ergebnisse zurückgeben indem du

die endgültige Höhe der Säule  $i$  in `finalHeight[i]` schreibst, für  $0 \leq i \leq n - 1$ .

## Subtasks

Bei allen Subtasks sind die Höhenangaben nichtnegative ganze Zahlen bis einschliesslich 100.000.

Subtask	Punkte	$n$	$k$	Bemerkung
1	8	$1 \leq n \leq 10.000$	$1 \leq k \leq 5.000$	keine zusätzlichen Limits
2	24	$1 \leq n \leq 100.000$	$1 \leq k \leq 500.000$	Alle Hinzufügephasen sind zeitlich vor allen Entfernpasen
3	29	$1 \leq n \leq 100.000$	$1 \leq k \leq 500.000$	keine zusätzlichen Limits
4	39	$1 \leq n \leq 2.000.000$	$1 \leq k \leq 500.000$	keine zusätzlichen Limits

## Implementierungsdetails

Du musst genau eine Datei abgeben, mit Namen `wall.c`, `wall.cpp` oder `wall.pas`. Diese Datei implementiert die oben beschriebene Funktion mit einer der folgenden Signaturen. Für C/C++ Programme musst du auch die Headerdatei `wall.h` einbinden.

### C/C++ Programm

```
void buildWall(int n, int k, int op[], int left[], int right[],  
int height[], int finalHeight[]);
```

### Pascal Programm

```
procedure buildWall(n, k : longint; op, left, right, height :  
array of longint; var finalHeight : array of longint);
```

## Sample -Grader

Der Sample-Grader liest die Inputdaten in folgendem Format:

- Zeile 1:  $n, k$ .
- Zeile  $2 + i$  ( $0 \leq i \leq k - 1$ ):  $op[i], left[i], right[i], height[i]$ .