Wall

Jian-Jia is building a wall by stacking bricks of the same size together. This wall consists of n columns of bricks, which are numbered 0 to $n-1$ from left to right. The columns may have different heights. The height of a column is the number of bricks in it.

Jian-Jia builds the wall as follows. Initially there are no bricks in any column. Then, Jian-Jia goes through k phases of *adding* or *removing* bricks. The building process completes when all k phases are finished. In each phase Jian-Jia is given a range of consecutive brick columns and a height h, and he does the following procedure:

- In an *adding* phase, Jian-Jia adds bricks to those columns in the given range that have less than h bricks, so that they have exactly h bricks. He does nothing on the columns having h or more bricks.
- In a *removing* phase, Jian-Jia removes bricks from those columns in the given range that have more than h bricks, so that they have exactly h bricks. He does nothing on the columns having h bricks or less.

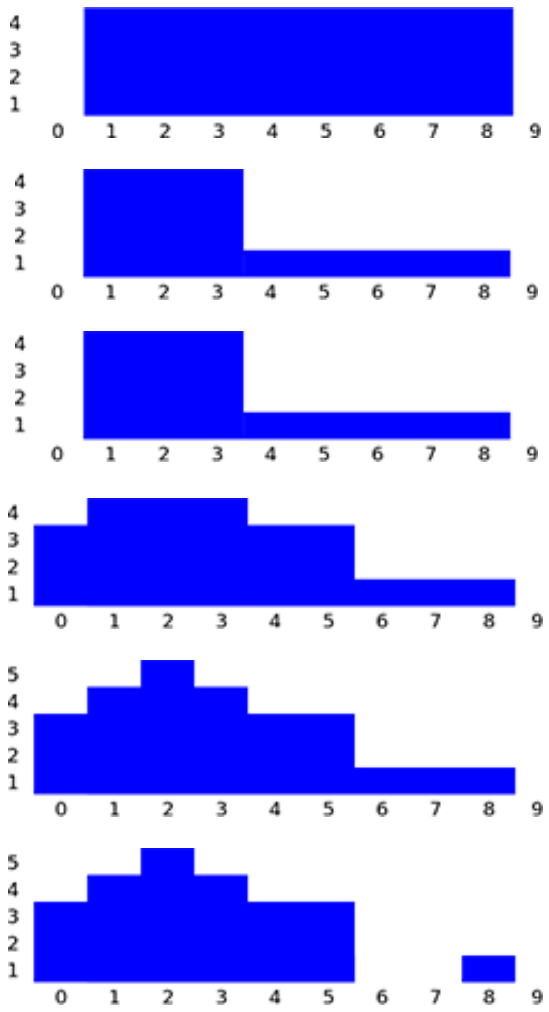
Your task is to determine the final shape of the wall.

Example

We assume that there are 10 brick columns and 6 wall building phases. All ranges in the following table are inclusive. Diagrams of the wall after each phase are shown below.

phase	type	range	height
0	add	columns 1 to 8	4
1	remove	columns 4 to 9	1
2	remove	columns 3 to 6	5
3	add	columns 0 to 5	3
4	add	column 2	5
5	remove	columns 6 to 7	0

Since all columns are initially empty, after phase 0 each of the columns 1 to 8 will have 4 bricks. Columns 0 and 9 remain empty. In phase 1, the bricks are removed from columns 4 to 8 until each of them has 1 brick, and column 9 remains empty. Columns 0 to 3, which are out of the given range, remain unchanged. Phase 2 makes no change since columns 3 to 6 do not have more than 5 bricks. After phase 3 the numbers of bricks in columns 0, 4, and 5 increase to 3. There are 5 bricks in column 2 after phase 4. Phase 5 removes all bricks from columns 6 and 7.



Task

Given the description of the k phases, please calculate the number of bricks in each column after all phases are finished. You need to implement the function `buildWall`.

- `buildWall(n, k, op, left, right, height, finalHeight)`
  - `n`: the number of columns of the wall.
  - `k`: the number of phases.
  - `op`: array of length k; `op[i]` is the type of phase i: 1 for an adding phase and 2 for a removing phase, for $0 \leq i \leq k-1$.
  - `left` and `right`: arrays of length k; the range of columns in phase i starts with column `left[i]` and ends with column `right[i]` (including both endpoints `left[i]` and `right[i]`), for $0 \leq i \leq k-1$. You will always have `left[i]` $\leq$ `right[i]`.
  - `height`: array of length k; `height[i]` is the height parameter of phase i, for $0 \leq i \leq k-1$.
  - `finalHeight`: array of length n; you should return your results by placing the final number of bricks in column i into `finalHeight[i]`, for $0 \leq i \leq n-1$.

Subtasks

For all subtasks the height parameters of all phases are nonnegative integers less or equal to $100,000$.

subtask	points	n	k	note
1	8	$1 \le n \le 10,000$	$1 \le k \le 5,000$	no additional limits
2	24	$1 \le n \le 100,000$	$1 \le k \le 500,000$	all adding phases are before all removing phases
3	29	$1 \le n \le 100,000$	$1 \le k \le 500,000$	no additional limits
4	39	$1 \le n \le 2,000,000$	$1 \le k \le 500,000$	no additional limits

Implementation details

You have to submit exactly one file, called `wall.c`, `wall.cpp` or `wall.pas`. This file implements the subprogram described above using the following signatures. You also need to include a header file `wall.h` for C/C++ program.

C/C++ program

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

Pascal program

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

Sample grader

The sample grader reads the input in the following format:

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