STATUE

With the policy of developing the Free Contest city into a famous travelling destination, many attractions has been constructed recently. The most famous among them are the statues located at the center of the city.

The statues are neatly placed into a rectangular table consisting of 2 rows and n columns (the columns are numbered from 0 to $n-1$). In the i-th column, the top row statue's height is a_i, while the bottom one's height is b_i. From the viewpoint of Free Contest's citizens, they consider the ugliness S of the attraction as the total value of the absolute height difference between each statue in the top row and each statue in the bottom row. Formally, the ugliness S can be defined as follow:

$$S = \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} |a_i - b_j|$$

To reduce the ugliness of the attraction (and thus make it more beautiful), the People's Committee of Free Contest city is planning to swap the position of some statues. However, due to limited budget, the commitee can only swap two statues that are in the same column. In other word, for each column i from 0 to $n-1$, the committee needs to decide to either swap the two statues in that column, or do nothing.

Your task is to help the committee by providing them a statue swapping plan so that the ugliness S of the attraction is minimized. If there are multiple plans that minimize the ugliness, you can report any of them.

Implementation Details

You should implement the following procedure:

```
int[] swap_statues(int[] a, int[] b)
```

- a, b: arrays of length n. For $0 \leq i < n$, a_i and b_i are the height of the top and bottom statue in column i, respectively.
- The procedure should return an array of length n (denote the array by t) representing the statue swapping plan. For $0 \leq i < n$, $t_i = 1$ if the committee need to swap the two statues in the column i, and $t_i = 0$ otherwise.
- The procedure is called exactly once.

Constraints

- $1 \leq n \leq 10^5$
- $1 \leq a_i, b_i \leq 10^8$ (for all $0 \leq i < n$)

Examples

Example 1

Consider the following call:

```
swap_statues([1, 3], [4, 5])
```

Initally, $a = [1, 3]$ and $b = [4, 5]$. The ugliness of the attraction is:

$$S = |1 - 4| + |1 - 5| + |3 - 4| + |3 - 5| = 3 + 4 + 1 + 2 = 10$$

An optimal plan is to swap statues in column 1. After swapping, $a = [4, 3]$ and $b = [1, 5]$. The ugliness become:

$$S = |4 - 1| + |4 - 5| + |3 - 1| + |3 - 5| = 3 + 1 + 2 + 2 = 8$$

This is the mimimum ugliness value possible in this example. Therefore, the `swap_statues` procedure can return $[1, 0]$.

Another valid plan that the procedure can return is $[0, 1]$.

Example 2

Consider the following call:

```
swap_statues([7, 9, 5], [8, 2, 4])
```

An optimal plan is to swap statues in column 0 and 1. After swapping, $a = [8, 2, 5]$ and $b = [7, 9, 4]$, and the ugliness become 27. Therefore, the `swap_statues` procedure can return $[1, 1, 0]$.

Another valid plan that the procedure can return is $[0, 0, 1]$.

Example 3

Consider the following call:

```
swap_statues([1, 2, 3, 4], [1, 2, 3, 4])
```

In this example, the two statues in each column have the same height, so the ugliness will not change regardless of the swapping plan. Therefore, the `swap_statues` procedure can return any valid array t (for example, $[1, 1, 0, 1]$).

Subtasks

1. (10 points) $n \leq 16$
2. (20 points) $a_{i-1} \leq a_i$, $b_{i-1} \geq b_i$ for all $1 \leq i < n$; $a_{n-1} \leq b_{n-1}$
3. (70 points) No additional constraints.

Sample Grader

The sample grader reads in the input in the following format:

- line 1: n
- line 2: $a_0\ a_1\ \ldots\ a_{n-1}$
- line 3: $b_0\ b_1\ \ldots\ b_{n-1}$

The sample grader prints your answers in the following format:

- line 1: $t_0\ t_1\ \ldots\ t_{n-1}$