CODEFORCES
Sponsored by TON

HOME TOP CATALOG CONTESTS GYM PROBLEMSET GROUPS RATING EDU API CALENDAR HELP

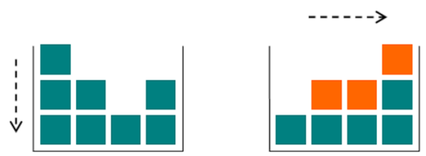
PROBLEMS SUBMIT CODE MY SUBMISSIONS STATUS HACKS ROOM STANDINGS CUSTOM INVOCATION

A. Gravity Flip

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little Chris is bored during his physics lessons (too easy), so he has built a toy box to keep himself occupied. The box is special, since it has the ability to change gravity.

There are n columns of toy cubes in the box arranged in a line. The i-th column contains a_i cubes. At first, the gravity in the box is pulling the cubes downwards. When Chris switches the gravity, it begins to pull all the cubes to the right side of the box. The figure shows the initial and final configurations of the cubes in the box: the cubes that have changed their position are highlighted with orange.



Given the initial configuration of the toy cubes in the box, find the amounts of cubes in each of the n columns after the gravity switch!

Input

The first line of input contains an integer n ($1 \le n \le 100$), the number of the columns in the box. The next line contains n space-separated integer numbers. The i-th number a_i ($1 \le a_i \le 100$) denotes the number of cubes in the i-th column.

Output

Output n integer numbers separated by spaces, where the i-th number is the amount of cubes in the i-th column after the gravity switch.

Examples

input Copy

```
4
3 2 1 2
```

output Copy

```
1 2 2 3
```

input Copy

```
3
2 3 8
```

output Copy

```
2 3 8
```

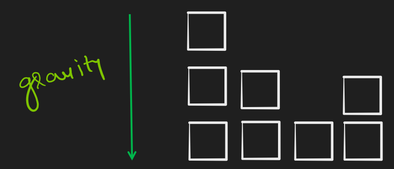
Note

The first example case is shown on the figure. The top cube of the first column falls to the top of the last column; the top cube of the second column falls to the top of the third column; the middle cube of the first column falls to the top of the second column.

N → 4

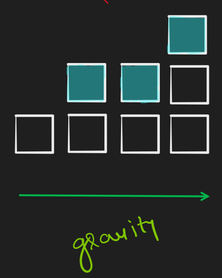
3	2	1	4

→ normal gravity array.



↓
also can be referred as the normal state.

3 displacements.



→ on observing carefully, we notice that the cubes are in a sorted fashion.

★ Goal:
→ Find no. of cubes that will be displaced after gravity switch.

Pseudocode:

```
Solve ( N, normal ) {
    vector<int> gravity_switch = normal;
    int displacements = 0;
    sort (gravity_switch);
    for (i ← 0 to N) {
        displacements += gravity_switch[i] - normal[i];
    }
    return displacements;
}
```

1. Duplicate the normal array into a new array called, let's say, gravity_switch.
2. Declare a variable called displacements. Initialize it to ZERO.
3. Sort the gravity_switch array.
4. Traverse the arrays.
5. displacements += gravity_switch[i] - normal[i]
6. Return displacements.