

Insertion Sort Part1

*Extracted from:*HR00002

Source file name: insertionsort1.py

Time limit: 1

Sorting

One common task for computers is to sort data. For example, people might want to see all their files on a computer sorted by size. Since sorting is a simple problem with many different possible solutions, it is often used to introduce the study of algorithms.

Insertion Sort

These challenges will cover *Insertion Sort*, a simple and intuitive sorting algorithm. We will first start with a nearly sorted list.

Insert element into sorted list

Given a sorted list with an unsorted number e in the rightmost cell, can you write some simple code to insert into the array so that it remains sorted?

Since this is a learning exercise, it won't be the most efficient way of performing the insertion. It will instead demonstrate the brute-force method in detail.

Assume you are given the array $arr = [1, 2, 4, 5, 3]$ indexed $\{0, 1, 2, 3, 4\}$. Store the value of $arr[4]$. Now test lower index values successively from 3 to 0 until you reach a value that is lower than $arr[4]$, $arr[1]$ in this case. Each time your test fails, copy the value at the lower index to the current index and print your array. When the next lower indexed value is smaller than $arr[4]$, insert the stored value at the current index and print the entire array.

The results of operations on the example array is:

Starting array:[1, 2, 4, 5, 3]

Store the value of $arr[4] = 3$. Do the tests and print interim results:

1 2 4 5 5

1 2 4 4 5

1 2 3 4 5

Input

There will be two lines of input:

The first line contains the integer n , the size of the array.

The next line contains n space-separated integers: $arr[0], arr[1], \dots, arr[n - 1]$.

The input must be read from standard input.

Output

Print the array as a row of space-separated integers each time there is a shift or insertion.

The output must be written to standard output.

Sample Input	Sample Output
5	2 4 6 8 8
2 4 6 8 3	2 4 6 6 8
	2 4 4 6 8
	2 3 4 6 8

Explanation

3 is removed from the end of the array.

In the 1st line $8 > 3$, so 8 is shifted one cell to the right.

In the 2nd line $6 > 3$, so 6 is shifted one cell to the right.

In the 3rd line $4 > 3$, so 4 is shifted one cell to the right.

In the 4th line $2 < 3$, so 3 is placed at position 1.

This problem was based by Hackerrank problem: "Insertion Sort - Part1" <https://www.hackerrank.com/challenges/insertionsort1/problem>.