Insertion Sort - Part 1

RATE THIS CHALLENGE

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 e 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 $0 \ldots 4$. 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
```

Function Description

Complete the *insertionSort1* function in the editor below. It should print the interim and final arrays, each on a new line.

insertionSort1 has the following parameter(s):

- *n*: an integer, the size of arr
- *arr*: an array of integers to sort

Input Format

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

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

Constraints

$1 \leq n \leq 1000$

$-10000 \leq arr[i] \leq 10000$

Output Format

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

Sample Input

```
5
2 4 6 8 3
```

Sample Output

```
2 4 6 8 8
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 1^{st} line $8 > 3$, so 8 is shifted one cell to the right.

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

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

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

Next Challenge

In the next Challenge, we will complete the insertion sort itself!