Insertion Sort - Part 2

In *Insertion Sort Part 1*, you sorted one element into an array. Using the same approach repeatedly, can you sort an entire unsorted array?

Guideline: You already can place an element into a sorted array. How can you use that code to build up a sorted array, one element at a time? Note that in the first step, when you consider an element with just the first element - that is already "sorted" since there's nothing to its left that is smaller.

In this challenge, don't print every time you move an element. Instead, print the array after each iteration of the insertion-sort, i.e., whenever the next element is placed at its correct position.

Since the array composed of just the first element is already "sorted", begin printing from the second element and on.

Input Format
There will be two lines of input:

- s - the size of the array
- ar - a list of numbers that makes up the array

Output Format
On each line, output the entire array at every iteration.

Constraints
$1 \leq s \leq 1000$
$-10000 \leq x \leq 10000, x \in ar$

Sample Input

```
6
1 4 3 5 6 2
```

Sample Output

```
1 4 3 5 6 2
1 3 4 5 6 2
1 3 4 5 6 2
1 3 4 5 6 2
1 2 3 4 5 6
```

Explanation
Insertion Sort checks 4 first and doesn't need to move it, so it just prints out the array. Next, 3 is inserted next to 1, and the array is printed out. This continues one element at a time until the entire array is sorted.

Task
The method **insertionSort** takes in one parameter: ar, an unsorted array. Use an Insertion Sort Algorithm to sort the entire array.