QuickSort 2 - Sorting

In the previous challenge, you wrote a *partition* method to split an array into 2 sub-arrays, one containing smaller elements and one containing larger elements. This means you 'sorted' half the array with respect to the other half. Can you repeatedly use *partition* to sort an entire array?

Guideline
In Insertion Sort, you simply went through each element in order and inserted it into a sorted sub-array. In this challenge, you cannot focus on one element at a time, but instead must deal with whole sub-arrays, with a strategy known as "divide and conquer".

When *partition* is called on an array, two parts of the array get 'sorted' with respect to each other. If *partition* is then called on each sub-array, the array will now be split into 4 parts. This process can be repeated until the sub-arrays are small. Notice that when partition is called on just two numbers, they end up being sorted.

Can you repeatedly call partition so that the entire array ends up sorted?

Print Sub-Arrays
In this challenge, print your array every time your partitioning method finishes, i.e. print every sorted sub-array The first element in a sub-array should be used as a pivot. Partition the left side before partitioning the right side. The pivot should not be added to either side. Instead, put it back in the middle when combining the sub-arrays together.

Input Format
There will be two lines of input:

- *n* - the size of the array
- *ar* - the *n* numbers of the array

Output Format
Print every partitioned sub-array on a new line.

Constraints
$1 <= n <= 1000$
$-1000 <= x <= 1000$, $x \in ar$
There are no duplicate numbers.

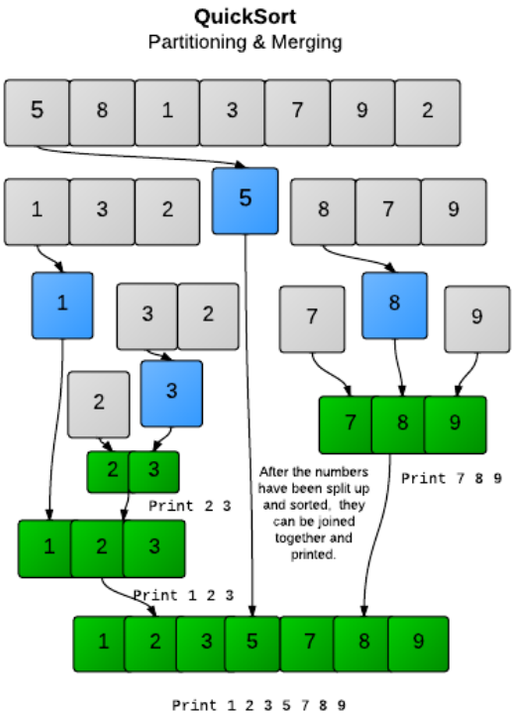
Sample Input

```
7
5 8 1 3 7 9 2
```

Sample Output

```
2 3
1 2 3
7 8 9
1 2 3 5 7 8 9
```

Explanation
This is a diagram of QuickSort operating on the sample array:



Task

The method **quickSort** takes in a parameter, **ar**, an unsorted array. Use the QuickSort Algorithm to sort the entire array.