Quick Sort 1 - Partition

Problem Statement

The previous challenges covered Insertion Sort, which is a simple and intuitive sorting algorithm. Insertion Sort has a running time of $O(N^2)$ which isn't fast enough for most purposes. Instead, sorting in the real-world is done with faster algorithms like Quicksort, which will be covered in the challenges that follow.

The first step of Quicksort is to partition an array into two smaller arrays.

Challenge
You're given an array *ar* and a number *p*. Partition the array, so that, all elements greater than *p* are to its right, and all elements smaller than *p* are to its left.

In the new sub-array, the relative positioning of elements should remain the same, i.e., if *n1* was before *n2* in the original array, it must remain before it in the sub-array. The only situation where this does not hold good is when p lies between *n1* and *n2*

i.e., *n1* > *p* > *n2*.

Guideline - In this challenge, you do not need to move around the numbers 'in-place'. This means you can create 2 lists and combine them at the end.

Input Format
You will be given an array of integers. The number *p* is the first element in *ar*.

There are 2 lines of input:

- *n* - the number of elements in the array *ar*
- the *n* elements of *ar* (including *p* at the beginning)

Output Format
Print out the numbers of the partitioned array on one line.

Constraints
1<=n<=1000
-1000<=x<= 1000 , x ∈ ar
All elements will be distinct

Sample Input

```
5
4 5 3 7 2
```

Sample Output

```
3 2 4 5 7
```

Explanation
p = 4. The 5 was moved to the right of the 4, 2 was moved to the left of 4 and the 3 is also moved to the left of 4. The numbers otherwise remained in the same order.

Task

Complete the method partition which takes an array *ar*. The first element in *ar* is the number *p*.