The previous challenges covered Insertion Sort, which is a simple and intuitive sorting algorithm with a running time of $O(n^2)$. In these next few challenges, we're covering a *divide-and-conquer* algorithm called Quicksort (also known as *Partition Sort*). This challenge is a modified version of the algorithm that only addresses partitioning. It is implemented as follows:

Step 1: Divide
Choose some pivot element, p, and partition your unsorted array, arr, into three smaller arrays: $left$, $right$, and $equal$, where each element in $left < p$, each element in $right > p$, and each element in $equal = p$.

For example: Assume $arr = [5, 7, 4, 3, 8]$
The pivot is at $arr[0] = 5$
arr is divided into $left = \{4, 3\}$, $equal = \{5\}$, and $right = \{7, 8\}$.
Putting them all together, you get $\{4, 3, 5, 7, 8\}$. Another valid solution is $\{3, 4, 5, 8, 7\}$.

Given arr and $p = arr[0]$, partition arr into $left$, $right$, and $equal$ using the *Divide* instructions above. Then print each element in $left$ followed by each element in $equal$, followed by each element in $right$ on a single line. Your output should be space-separated and does not have to maintain ordering of the elements within the three categories.

Function Description

Complete the *quickSort* function in the editor below. It should return an array of integers as described above.

quickSort has the following parameter(s):

- *arr*: an array of integers where $arr[0]$ is the pivot element

Input Format

The first line contains n, the size of the array arr.
The second line contains n space-separated integers describing arr (the unsorted array). The first integer (corresponding to $arr[0]$) is your pivot element, p.

Constraints

- $1 \leq n \leq 1000$
- $-1000 \leq arr[i] \leq 1000$ where $0 \leq i < n$
- All elements will be unique.

Output Format

On a single line, print the partitioned numbers (i.e.: the elements in $left$, then the elements in $equal$, and then the elements in $right$). Each integer should be separated by a single space.

Sample Input

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

Sample Output

```
3 2 4 5 7
```

Explanation

$arr = [4, 5, 3, 7, 2]$ *Pivot*: $p = arr[0] = 4$.
$left = \{\}$; $equal = \{4\}$; $right = \{\}$

$arr[1] = 5 > p$, so it's added to $right$.
$left = \{\}$; $equal = \{4\}$; $right = \{5\}$

$arr[2] = 3 < p$, so it's added to $left$.
$left = \{3\}$; $equal = \{4\}$; $right = \{5\}$

$arr[3] = 7 > p$, so it's added to $right$.
$left = \{3\}$; $equal = \{4\}$; $right = \{5, 7\}$

$arr[4] = 2 < p$, so it's added to $left$.
$left = \{3, 2\}$; $equal = \{4\}$; $right = \{5, 7\}$

We then print the elements of $left$, followed by $equal$, followed by $right$, we get: `3 2 4 5 7`.

You don't need to maintain ordering, so another valid solution would be `2 3 4 5 7`.