




# Challenge: Implement Partition


The partition function should partition the subarray `array[p..r]` so that all elements in `array[p..q-1]` are less than or equal to `array[q]` (the pivot) and all elements in `array[q+1..r]` are greater than `array[q]`, and it returns the index `q` of where the pivot ends up.

Use the provided `swap()` function for swapping.

 Java

 Python

 C++

 JS

```
class Solution {
    // Swaps two items in an array, changing the original array
    static void swap(int[] array, int firstIndex, int secondIndex) {
        int temp = array[firstIndex];
        array[firstIndex] = array[secondIndex];
        array[secondIndex] = temp;
    };

    public static void partition(int[] array, int p, int r) {
        // Compare array[j] with array[r], for j = p, p+1,...r-1
        // maintaining that:
        //  array[p..q-1] are values known to be <= to array[r]
        //  array[q..j-1] are values known to be > array[r]
        //  array[j..r-1] haven't been compared with array[r]
        // If array[j] > array[r], just increment j.
        // If array[j] <= array[r], swap array[j] with array[q],
        //   increment q, and increment j.
        // Once all elements in array[p..r-1]
        //   have been compared with array[r],
        //   swap array[r] with array[q], and return q.

    }
}
```

