1. The difference between my algorithm and quicksort is that my algorithm does not sort both sides of the array. It starts by performing a single quicksort with the pivot being the last element of the array, and then looks at the pivot index (pi). If the pivot index is equal to k (k-1 to convert to base 0 index), then that means the value that is at the pivot index is the kth smallest value, as there are k-1 values to the left of it. If there are more than k elements to the left of the pivot (i.e pi > k), this means that the kth smallest value is somewhere in the left half of the semi-sorted array, so quicksort is again called on only the left side. If there are less than k elements to the left of k (i.e pi<k), this means that the kth smallest element is somewhere in the right half of the array, so a recursive call is made, passing only the right half of the array. This way, you only have to sort one half of the array, and you only have to sort until the pivot index equals k. This improves on standard quicksort, as standard quicksort would have to fully sort the array and index over it to the kth index.
2. The base case of my recursive function is when the pivot index equals k-1. At this point, no further recursive calls are made and the array ceases to be modified.