Quick Sort (on AP Test)

1. If the size of the array is not larger than 1
    a. Assume an array of size 1 or 0 is in order.
2. If the size of the array is larger than 1
    a. Pick an element in the array to act as the "pivot".
    b. Reorder the array so that all the elements that are less than the pivot are located to the left of it, and all those that are greater than it are located to the right of it. After this reorder, the pivot element will be located in its correct position in the final sorted array.
    c. Recursively call the algorithm on both the "greater than pivot" and "less than pivot" parts of the array.

Reordering

The reordering step is a big part of the algorithm. There are several ways to do it, but one way is to:

1. Take your pivot element and swap it with the end element while you reorder
2. Have an finalIndex variable that will hold the final location of the pivot when the reorder method is done. Initialize it to the start of the array.
3. Loop from the beginning of the array until the end.
   Each time you find an element less than your pivot:
    a. swap it with the element at your finalIndex variable
    b. increment finalIndex
4. Swap finalIndex with the last element (which is holding your pivot.

Quick Sort Algorithm

```
public static void quickSort(int[] array, int start, int end)
{
     //if there are still elements to sort
     if(start<end)
     {
          //pick the partition and rearrange the array around it.
          int index = reorder(array, start, end);

          //recursively quick sort the two halves of the array
          quickSort(array, start, index-1);
          quickSort(array, index+1, end);
     }
}

public static int reorder(int[] array, int start, int end)
{
     //pick the index of your pivot element. This can vary,
     //but for now we'll use the middle element.
     int index = (start+end)/2;
     int pivotValue = array[index];

     //move our pivot element to the end of the array while we reorder
     swap(array, index, end);

     //index will hold the final place for the pivot in the
     //reordered array. For now, put it as the first element.
     index = start;
     for(int i = start; i<end; i++)
     {
          //if you find an element less than pivot, swap it with
          //the element at index, then increment index because the
          //final position of our pivot has now increased by one
          if(array[i]<=pivotValue) {
               swap(array, index, i);
               index++;
          }
     }

     //put pivot in its correct spot and return its position
     swap(array, index, end);
     return index;
}

//simple swap method
public static void swap(int[] array, int pos1, int pos2}
{
     int tmp = array[pos1];
     array[pos1] = array[pos2];
     array[pos2] = tmp;
}
```