**public void** quickSort(**int** array[])

// pre: array is full, all elements are non-null integers

// post: the array is sorted in ascending order

{

quickSort(array, 0, array.length - 1); // quicksort all the elements in the array

}

**public void** quickSort(**int** array[], **int** start, **int** end)

{

**int** bot = start; // index of left-to-right scan

**int** top = end; // index of right-to-left scan

**if** (end - start >= 1) { // check that there are at least two elements to sort

**int** pivot = array[start]; // set the pivot as the first element in the partition

**while** (top > bot) { // while the scan indices from left and right have not met,

**while** (array[bot] <= pivot && bot<=end && top > bot) // from the left, look for the first

bot++; // element greater than the pivot

**while** (array[top] > pivot && top >= start && top >= bot) // from the right, look for the first

top--; // element not greater than the pivot

**if** (top > bot) // if the left seekindex is still smaller than

swap(array, bot, top); // the right index, swap the corresponding elements

}

swap(array, start, top); // after the indices have crossed, swap the last element in

// the left partition with the pivot

quickSort(array, start, top - 1); // quicksort the left partition

quickSort(array, top + 1, end); // quicksort the right partition

}

**else** { // if there is only one element in the partition, do not do any sorting

**return;** // the array is sorted, so exit

}

}

**public void** swap(**int** array[], **int** index1, **int** index2)

// pre: array is full and index1, index2 < array.length

// post: the values at indices 1 and 2 have been swapped

{

**int** temp = array[index1]; // store the first value in a temp

array[index1] = array[index2]; // copy the value of the second into the first

array[index2] = temp; // copy the value of the temp into the second

}