Pseudocode for Lab 2

*MergeSort Class*

**Algorithm sort(array):**

**Input:** an integer array

**Output:** a sorted array in ascending order

If (array length < 2) then return as there are not enough values in the array to sort

For (index < midpoint of the array)

Left sub array is equal to the left half of the input array

For (index starting at midpoint of the array < length of the array)

Right sub array is equal to the right half of the input array

Sort the left sub array until its size is less than 2

Sort the right sub array until its size is less than 2

Merge the left and right sub array

**Algorithm merge(left sub array, right sub array, original array):**

**Input:** an integer array containing the left half of the original input array, an integer array containing the right half of the original input array, the original array the two sub arrays came from.

**Output:** a merged, sorted array of the left and right input sub arrays

While (left sub array index is less than the size of left sub array and right sub array index is less than the size of the right sub array)

If (left sub array value at left index is less than or equal to right sub array value at righ index)

Merged array value at merge index is equal to left sub array value at left index

Else

Merged array value at merge index is equal to right sub array value at right index

While (left sub array index is less than the size of the left sub array)

Merged array value at merge index is equal to left sub array value at left index

While (right sub array index is less than the size of the right sub array)

Merged array value at merge index is equal to right sub array value at right index

*QuickSort Class*

**Algorithm sort(array, lower index, upper index):**

**Input:** integer array to be sorted, lower index of the array to sort from, upper index of array to sort up to

**Output:** a sorted array in ascending order

If (array size is less than two) return because there aren’t enough values in the array to sort

If (lower index is less than the upper index)

Partition the array at the upper index and sort the array based on that partition index

Sort the array from the lower index up until the partition index

Sort the array from the partition index up to the upper input index

**Algorithm partition(array, lower index, upper index):**

**Input:** Integer array that is being sorted, lower index of the array, upper index of the array

**Output:** a partition index to sort the array in two sub arrays

For (index starting at the lower index up to upper index)

If (array at index is less than or equal to the pivot point)

Swap the values at the two indexes

Swap the values in the array at the upper index and current index

Return the partition index