Mergesort and Quicksort Pseudocode

ALGORITHM MERGE(B, C, A)
//Input: a sorted array $B[0..p-1]$
//Input: a sorted array $C[0..q-1]$
//Output: a sorted array $A[0..(p+q-1)]$
$i \leftarrow 0$
$j \leftarrow 0$
$k \leftarrow 0$
// take smallest item from B and C and put into A
while $i < p$ **and** $j < q$ **do**
  if $B[i] < C[j]$ **then**
    $A[k] \leftarrow B[i]$
    $i \leftarrow i + 1$
  else
    $A[k] \leftarrow C[j]$
    $j \leftarrow j + 1$
  $k \leftarrow k + 1$
// if items remain in B, put them in A
while $i < p$ **do**
  $A[k] \leftarrow B[i]$
  $i \leftarrow i + 1$
  $k \leftarrow k + 1$
// if items remain in C, put them in A
while $j < q$ **do**
  $A[k] \leftarrow C[j]$
  $j \leftarrow j + 1$
  $k \leftarrow k + 1$

ALGORITHM MERGESORT(A)
//Input: an array $A[0..n-1]$
if $n > 1$ **then**
  copy first half of array A into a temp array B
  copy second half of array A into a temp array C
  $MergeSort(B)$
  $MergeSort(C)$
  $Merge(B, C, A)$

ALGORITHM QUICKSORT(A, lt, rt)
//Input: an array $A[0..n-1]$
//Input: lower rt and upper rt bounds of the subarray to sort
//The initial call would be with $lt = 0$ and $rt = n - 1$
if $lt < rt$ **then**
$s \leftarrow Partition(A, lt, rt)$
// s is pivot element location
$QuickSort(A, lt, s-1)$
$QuickSort(A, s+1, rt)$

ALGORITHM PARTITION(A, lt, rt)
//Input: an array $A[0..n-1]$
//Input: lower lt and upper rt bounds of the subarray
// to partition
$p \leftarrow A[lt]$ // select pivot
$i \leftarrow lt$
$j \leftarrow rt + 1$
repeat
repeat
$i \leftarrow i + 1$
until $i = rt$ **or** $A[i] \geq p$
repeat
$j \leftarrow j - 1$
until $j = lt$ **or** $A[j] \leq p$
$swap(A[i], A[j])$
until $i \geq j$
$swap(A[i], A[j])$ // undo last swap
$swap(A[lt], A[j])$ // place pivot
return j