Exercise 1

1. Run the MERGE-SORT algorithm on the following array of characters $A = \langle c, w, x, m, o, z, q, e \rangle$. Give the state of the array after five calls to MERGE have been performed.
2. State a loop invariant for the **while** loop of lines 12-18 of the MERGE procedure (from the book). Show how to use it along with the **while** loops of lines 20-23 and 24-27, to prove that the MERGE procedure is correct. (CLRS 2.3-3)
3. You can also think of insertion sort as a recursive algorithm. In order to sort $A[1 : n]$, recursively sort the subarray $A[1 : n - 1]$ and then insert $A[n]$ into the sorted subarray $A[1 : n-1]$. Write pseudo-code for thise recursive version of insertion sort. Give a recursion for its worst-case running time (might be easier after the second part of the lecture). (CLRS 2.3-5)

Exercise 2

1. Using Figure 7.1 as a model, illustrate the operation of PARTITION on the array $A = \langle 13, 19, 9, 5, 12, 8, 7, 4, 21, 2, 6, 11 \rangle$. (CLRS 7.1-1)
2. Give a brief argument that the running time of PARTITION on a subarray of size n is $\Theta(n)$. (CLRS 7.1-3)
3. Show that the running time of QUICKSORT is $\Theta(n^2)$ when the array A contains distinct elements and is sorted in decreasing order. (CLRS 7.2-3)

Exercise 3

Consider the problem of finding the smallest element in a nonempty array of numbers $A[1 : n]$.

1. Write an *incremental* algorithm that solves the above problem and determine its asymptotic worst-case running time.
2. Write an *divide-and-conquer* algorithm that solves the above problem and determine its asymptotic worst-case running time.
3. Assume that the length of A is a power of 2. Write a recurrence describing how many comparison operations (among elements of A) your divide-and-conquer algorithm performs (might be easier after the second part of the lecture).