Lab 4 Exercise 1 Questions

2.

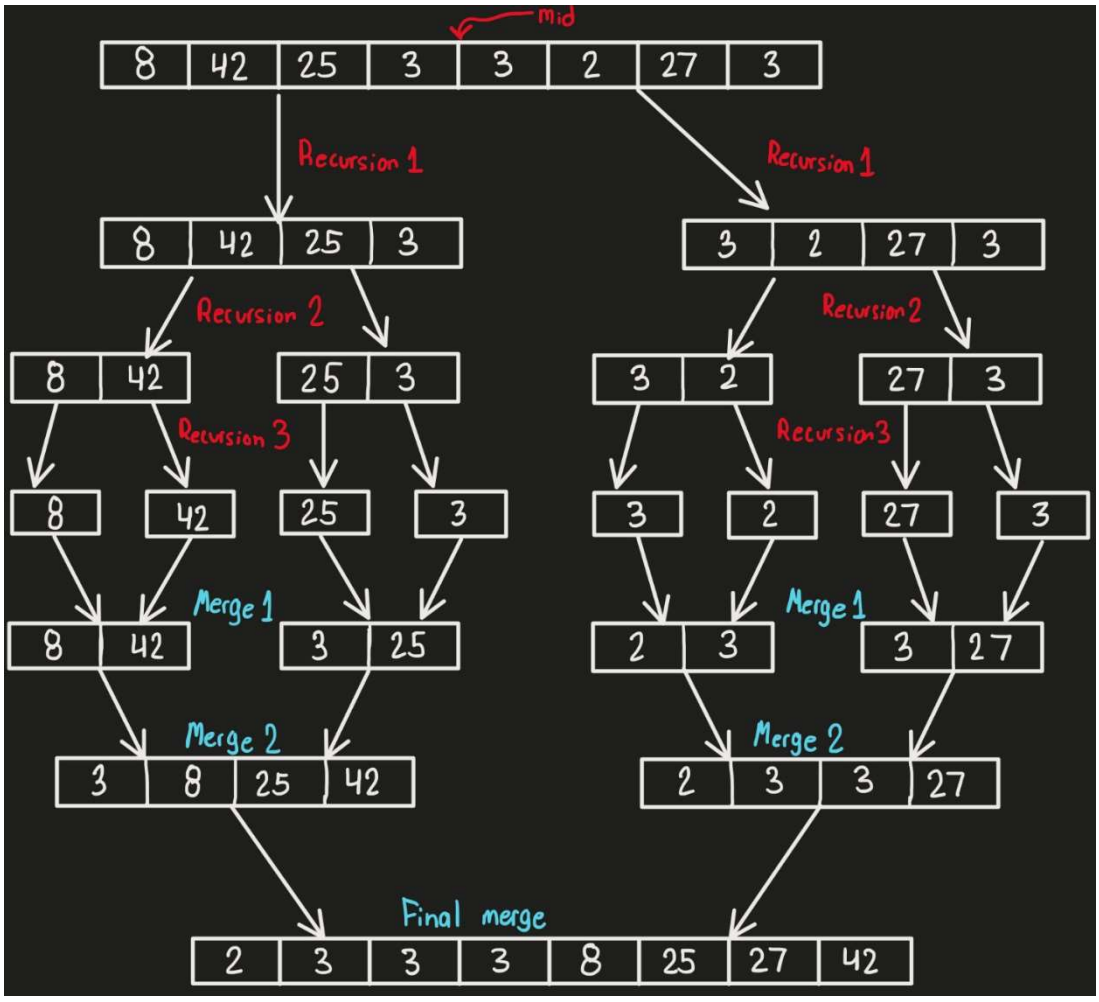
In order to argue that worst-case complexity of the merge sort algorithm is O(nlogn) ill split the algorithm into its divide and conquer components.

The divide components comes from the merge_sort function. The function will recursively divide the array into halves until the base case (low < high) is reached. That step has a complexity of O(log n) because the array is recursively divided into halves until it reaches a size of 1 and can no longer be divided.

The conquer component comes from the merge function. Once the array can no longer be divided into halves, the function will put it back together while also ordering each element. The merge function has a complexity of O(n) since each element needs to be checked.

Since the merge_sort function has a complexity of O(log n) and the merge function has a complexity of O(n), the overall complexity of the algorithm is O(nlogn)

3. (still needs some kind of discussion)



4.

The number of steps seen in the manual application of the merge sort algorithm is consistent with the complexity analysis discussed in question 2. At each recursion level in the manual application, the array is split in half until it reached the base case; thus the number of recursions required is logarithmic in the size of the input array.

Again, as discussed in question 2 the complexity analysis of divide component was O(logn) and in the manual application the amount of steps aligns with that. The same is said for the conquer component as well.