Part 4

Q1:

The worst and best case Big-Oh complexity of merge sort is **O(nlogn).** This is because in both cases whether the array is sorted or not, it would still need to divide the array into two halves until one element is left at the end. The array follows the same steps for both scenario and any comparisons are not skipped if array is already sorted. This sort works differently unlike other sorting algorithms. This is why it has O(nlogn) complexity.

Q2:

The worst and best complexity of merge Sort working iteratively is **O(nlogn).** It is the same as it was for recursive. This is because it is following the same divide and conquer approach. It is still dividing the array into two parts until one element is left in an array. Hence, the complexity will be the same for both approaches, i.e: iterative and recursive.