Question - 1

SCORE: **5 points**

What does MergeSort have in common with QuickSort

Using your knowledge of MergeSort and QuickSort, choose which of the following statements is true.

- (●) They each tend to be O(n log n) in the average case; they are both recursive, operating on two or more partitions.
- ( ) They are both "stable"sorts
- ( ) They both use a lot of extra memory
- ( ) Unlike Quicksort, MergeSort can benefit from having a recursion cutoff which uses InsertionSort.

Question - 2

SCORE: **5 points**

Minimum number of compares

What is the theoretical minimum number of compares that a sort algorithm must perform when data is randomly shuffled prior to the sort?

- ( ) n!
- (●) n log n
- ( ) n2
- ( ) n

Question - 3

SCORE: **5 points**

Quick Sort

Given the following list of numbers [14, 17, 13, 15, 19, 10, 3, 16, 9, 12] which answer shows the contents of the list after the second partitioning according to the quicksort algorithm? (Hint : The first partition step works on the complete array, whereas the second partition step just works on the left half of the array)

- ( ) [9, 3, 10, 13, 12]
- ( ) [9, 3, 10, 13, 12, 14]
- ( ) [9, 3, 10, 13, 12, 14, 17, 16, 15, 19]
- (●) [9, 3, 10, 13, 12, 14, 19, 16, 15, 17]

Question - 4

SCORE: **5 points**

Quick Sort comparisons

Given two arrays of equal length, where A1 is the array [1, 2, 3, 4, 5] and A2 is [4, 1, 5, 3, 2], which is true about the number of comparisons performed during quicksort if the shuffle step is omitted? Let C1 be the number of comparisons performed in sorting A1, and C2 be the number of comparisons performed in sorting A2. Assume that the first element in any sub-array is used as the pivot.

- ○ C1==C2
- ○ C1<C2
- ● C1>C2

Question - 5

SCORE: **30 points**

Quick Sort Implementation

Given a list of elements of generic type '*T*' (which extends *Comparable*<*T*>) , please implement the sort and partition methods for QuickSort that will ensure that *List*<*T*> is sorted.