Sorting Part 2: Recursive Sorts

Your boss was super impressed with your last library. It allowed him to choose which sort he wanted to use at any given moment… that is until he heard about recursive sorts. While he knew what “sorting” was, he had never heard the word “recursive” before, which meant that he had to have these new sorts RIGHT AWAY!

Your boss wants you to update your library to include MergeSort and QuickSort functions.

# Requirements

1. You will build upon your library from the previous sorting lab.
2. Since you are adding methods to an existing library, it is expected that all previous requirements (including names of methods and namespaces/packages) have been met and remain unchanged.
3. To your Sorter class, add the following methods:
   1. MergeSort  
      QuickSort
   2. Each method is required to implement the sorting algorithm of the same name.
   3. Use proper casing for your chosen language in naming your methods.
   4. Each method will take in a generic array as a parameter, but it must be constrained to datatypes that implement the IComaprable<T> interface (C#) or Comparable<T> interface (Java).
   5. Each method returns void as each sort is expected to be “in place”.
4. Your deliverable is the DLL or JAR itself, NOT the code base.

# Rubric

**Automatic Zero:** Your deliverable is not a library, any exception is thrown by your code, or you fail to correctly implement any of the expected sorting algorithms.

(15 points) MergeSort works

(15 points) QuickSort works