Project I: Comparison-based Sorting Algorithms

You will implement some comparison-based sorting algorithms and observe their performance for different size.

Implement the following sorting algorithms.

1. Insertion sort
2. Merge sort
3. In-place quicksort (any random item or the first or the last item of your input can be pivot).
4. Modified quicksort
   a. Use median-of-three as pivot.
   b. For small subproblem size (≤ 10), you must use insertion sort.

Execution instructions:

1. Run these algorithms for different input sizes (e.g. n = 500, 1000, 2000, 4000, 5000, 10000, 20000, 30000, 40000 and 50,000). You will randomly generate numbers for your input array. Record the execution time (need to take the average as discussed in the class) in a table and later plot them all in a single graph against input size. Note that you will compare these sorting algorithms for the same data set.
2. In addition, observe and present performance of the following two special cases:
   a. Input array is already sorted.
   b. Input array is inversely sorted.

Grading Scheme:

Item	Points
Insertion-sort implementation	10
Merge-sort implementation	10
In-place quicksort	15
Modified quicksort	25
Simulation (instruction 1) – output and figure	30
Special cases (instruction 2)	10

Submission instructions:

1. You can work in a team of size TWO.
2. A well-formatted report covering results, figure and your understanding. At the end of the report, include code of each sorting method.
3. In addition, you must upload code so that I can run and check if necessary.
4. You can use any programming language (e.g. C/C++, Java, Python, Haskell, etc) of your choice.