COMP 5112. Lab 5: Sorting

Question 1. Quicksort

Please **fill** in the following missing parts in the file `Quicksort.java`, then **run** the program.

```
// 1. _____________________________
// 2. _____________________________
// 3. _____________________________
```

The correct output when running the program output should contain:

```
*** Input: 10 7 1 6 9 3 2 4 8 5
*** Output: 1 2 3 4 5 6 7 8 9 10
```

The answer is inside Q1Answer.zip (available at Blackboard)

Question 2. The running time of Quicksort

In `Quicksort.java`, we can generate the worst case input and the average case input by calling `generateWorstCaseInput` and `generateAverageCaseInput`, respectively.

Please **modify** the line for generating the array A in main, **run** the program, and then **fill** in the following table.

Input size	*Average case input: running time (steps)*	*Worst case input: running time (steps)*
10	27	54
100	797	5049
1000	11460	500499

What are your observations on the above results?

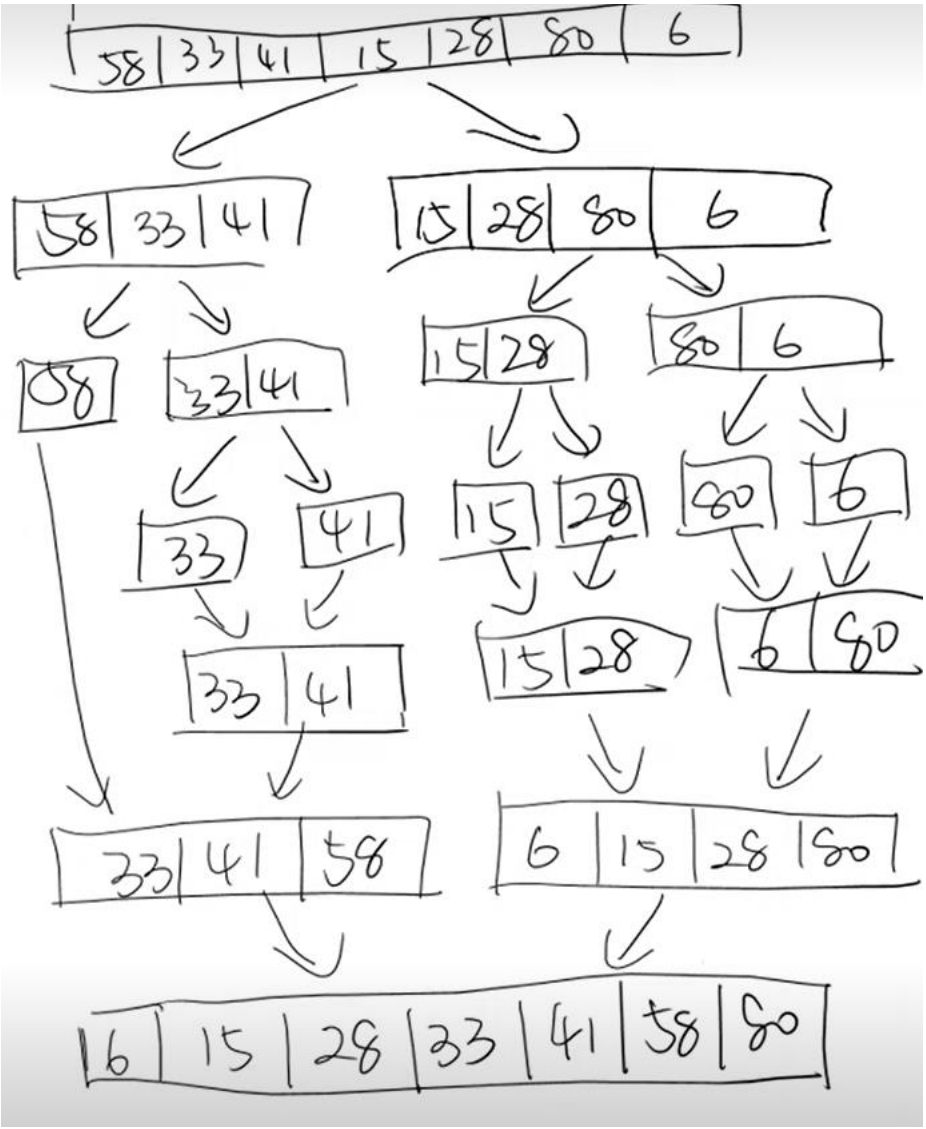
You can uncomment line 154 – 160 in QuickSort.java to run this experiment.

We can find that when the input size increases, the running time grows at a rate roughly consistent with the average and worst time complexity of QuickSort, i.e., O(nlogn) and O(n^2).

Question 3.

Show the running steps of the merge-sort algorithm on the following array:

	0	1	2	3	4	5	6
A	58	33	41	15	28	80	6



(screenshot from the lab video)

Question 4.

Find the worst-case input at n=5
for the following Insertion-Sort algorithm:

```
Insertion-Sort ( Array A[0..n-1] )
1. i ← 1
2. while i < n
3.     j ← i
4.     while j > 0 and A[j-1] > A[j]
5.         swap A[j] and A[j-1]
6.         j ← j-1
7.     i ← i + 1
```

Any input array of size 5 that is strictly decreasing is the worst-case input for Insertion-Sort algorithm. For example:
Input: 5 4 3 2 1