COP 3502 Study Group Sheet: Sorting Solutions

1) Show the state of the following array after each iteration of Bubble Sort of the array shown in the first row of the chart.

Original	12	3	19	5	4	13	18	16	7
1st pass	3	12	5	4	13	18	16	17	19
2nd pass	3	5	4	12	13	16	17	18	19
3rd pass	3	4	5	12	13	16	17	18	19
4th pass	3	4	5	12	13	16	17	18	19
5th pass	3	4	5	12	13	16	17	18	19
6th pass	3	4	5	12	13	16	17	18	19
7th pass	3	4	5	12	13	16	17	18	19
8th pass	3	4	5	12	13	16	17	18	19

2) Show the state of the following array after each iteration of Selection Sort of the array shown in the first row of the chart.

Original	12	3	19	5	4	13	18	16	7
1st pass	12	3	7	5	4	13	18	16	19
2nd pass	12	3	7	5	4	13	16	18	19
3rd pass	12	3	7	5	4	13	16	18	19
4th pass	12	3	7	5	4	13	16	18	19
5th pass	4	3	7	5	12	13	16	18	19
6th pass	4	3	5	7	12	13	16	18	19
7th pass	4	3	5	7	12	13	16	18	19
8th pass	3	4	5	7	12	13	16	18	19

3) Show the state of the following array after each iteration of Insertion Sort of the array shown in the first row of the chart.

Original	12	3	19	5	4	13	18	16	7
1st pass	3	12	19	5	4	13	18	16	7
2nd pass	3	12	19	5	4	13	18	16	7
3rd pass	3	5	12	19	4	13	18	16	7
4th pass	3	4	5	12	19	13	18	16	7
5th pass	3	4	5	12	13	19	18	16	7
6th pass	3	4	5	12	13	18	19	16	7
7th pass	3	4	5	12	13	16	18	19	7
8th pass	3	4	5	7	12	13	16	18	19

4) Show the state of the following array after the 1st Merge completes, the 2nd Merge completes, the 3rd Merge completes, …, 7th Merge completes:

Original	12	3	19	5	4	13	18	16
1st Merge	3	12	19	5	4	13	18	16
2nd Merge	3	12	5	19	4	13	18	16
3rd Merge	3	5	12	19	4	13	18	16
4th Merge	3	5	12	19	4	13	18	16
5th Merge	3	5	12	19	4	13	16	18
6th Merge	3	5	12	19	4	13	16	18
7th Merge	3	4	5	12	13	16	18	19

5) Consider running the in place Partition algorithm show in class on the following array, with the partition element initially stored in index 0. Show the state of the array right after the partition is complete:

Original	12	6	19	40	13	4	5	38	7
After Part	4	6	7	5	12	13	40	38	19

6) Explain why Quick Sort can be "done in place", while Merge Sort can not.

The partition only uses one temp variable to swap pairs of items out of order, but merging two sorted arrays can't be done in this fashion and is much more easily done by copying values into a new array and then copying them back to the original array.

7) Why is the worst case run time of Quick Sort $O(n^2)$, but the worst case run time of Merge Sort is O(nlgn)?

Merge Sort will always split its arrays to sort recursively into two halves. Due to this equal split, the recurrence relation is fixed and guarantees the O(nlgn) run time. Quick Sort can split the input array in anyway. The worst case split is one array of size n-1 and another array of size 0, when the input array is size n. Repeatedly getting this split creates a run time of $O(n^2)$.

8) Why is Quick Sort faster in practice than Merge Sort, even though theoretically, in the worst case Merge Sort is faster?

Because it can be done in place and doesn't have the overhead of copying values into temporarily allocated arrays and copying the values back.