1. Complete using Bubble sort to sort MyList in descending order

Iteration #	MyList	Swap #	Comparison #
0	[23, -12, -15, 10, 45, 3, 13, -19]	X	X
1	[23, -12, 10, 45, 3, 13, -15, -19]	4	7
2	[23, 10, 45, 3, 13, -12, -15, -19]	4	6
3	[23, 45, 10, 13, 3, -12, -15, -19]	2	5
4	[45, 23, 13, 10, 3, -12, -15, -19]	2	4
5	[45, 23, 13, 10, 3, -12, -15, -19]	0	3
6	[45, 23, 13, 10, 3, -12, -15, -19]	0	2
7	[45, 23, 13, 10, 3, -12, -15, -19]	0	1
8	[45, 23, 13, 10, 3, -12, -15, -19]	0	0

2. Complete using Selection sort to sort MyList in ascending order

Iteration #	MyList	Swap #	Comparison #
0	[23, -12, -15, 10, 45, 3, 13, -19]	X	X
1	[23, -12, -15, 10, -19, 3, 13, 45]	1	8
2	[13, -12, -15, 10, -19, 3, 23, 45]	1	7
3	[3, -12, -15, 10, -19, 13, 23, 45]	1	6
4	[3, -12, -15, -19, 10, 13, 23, 45]	1	5
5	[-19, -12, -15, 3, 10, 13, 23, 45]	1	4
6	[-19, -15, -12, 3, 10, 13, 23, 45]	1	3
7	[-19, -15, -12, 3, 10, 13, 23, 45]	0	2

3. Complete using Insertion sort to sort MyList in descending order

Iteration #	MyList	Swap #	Comparison #
0	[23, -12, -15, 10, 45, 3, 13, -19]	X	X
1	[23, -12, -15, 10, 45, 3, 13, -19]	0	1
2	[23, -12, -15, 10, 45, 3, 13, -19]	0	1
3	[23, 10, -12, -15, 45, 3, 13, -19]	(it shifts 2)	3

4	[45, 23, 10, -12, -15, 3, 13, -19]	(it shifts 4)	4
5	[45, 23, 10, 3, -12, -15, 13, -19]	(it shifts 2)	3
6	[45, 23, 13, 10, 3, -12, -15, -19]	(it shifts 4)	5
7	[45, 23, 13, 10, 3, -12, -15, -19]	0	1

Shifting is different than swapping and insertion does not swap. Instead, according to the textbook, it shifts which take approximately a third of the processing power of a swap. So for the swap box, I included the number of shifts because I did not know what should be used and just put one in the place of swaps.

4. Complete using Merge sort to sort MyList in ascending order

Recursive Call #	Left Sub-List	Right Sub-List	Swap #	Comparison #	Post Merge
1	[23, -12, -15, 10]	[45, 3, 13, -19]		7	[-19, -15, -12, 3, 10, 13, 23, 45] done
2	[23,-12]	[-15,10]		3	[-15,-12,10,23] Return to call 1
3	[23]	[-12]		1	[-12,23] Return to call 2
4	[-15]	[10]		1	[-15,10] Return to call 2
5	[45,3]	[13,-19]		3	[-19,3,13,45] Return to call 1
6	[45]	[3]		1	[3,45] Return to call 5
7	[13]	[-19]		1	[-19,13] Return to call 5

No swapping only merging, but it uses temporary space and does not sort in place

5. Complete using Quicksort to sort MyList in descending order

Recursive Call #	MyList	Pivot Value	Swap #	Comparison #
0	[23, -12, -15, 10, 45, 3, 13, -19]			
1	[45,23,-15,10,-12,3,13,-19]	23	2	7

2 --- 3 --- 4 --- 5 --- 6 --- 7	[45, 23, 13, 10, -12, 3, -15, -19] --- [45, 23, 13, 10, -12, 3, -15, -19] --- [45, 23, 13, 10, -12, 3, -15, -19] --- [45, 23, 13, 10, 3, -12, -15, -19] --- [45, 23, 13, 10, 3, -12, -15, -19] --- [45, 23, 13, 10, 3, -12, -15, -19]	-15 --- 13 --- 10 --- -12 --- -15 --- -19	1 --- 0 --- 0 --- 1 --- 0 --- 0	5 --- 5 --- 4 --- 3 --- 1 --- done

6. Complete the table below. Which sorting technique is most efficient? Why? Give a detailed explanation for your answer

	Total Iterations/Calls	**Total Comparisons**	**Total Swaps**
Bubble Sort	8	28	12
Selection Sort	7	35	6
Insertion Sort	7	18	(12 shifts) no swaps
Merge Sort	7	17	No swaps Just merging
Quick Sort	7	25	4

Quicksort is most efficient usually because it has fewer swaps and sorts in place without creating temporary space as merge sort does. However, it did run into more comparisons than necessary because some values were already in the correct position. Also in most cases, it is n log n time. Which is better than the average cases of bubble, selection, and insertion with n^2 average cases. Merge sort has the same average as Quicksort, but quicksort does not have to use additional space. Therefore, quicksort is more efficient generally.

But in the case of this list, if you don't mind the additional space merge sort uses, then Merge Sort is most efficient. This is because it makes less swaps and comparisons than Quicksort. Both sorting algorithms average n log n time but Quick sorts worst case scenario (n^2) is worse than Merge sort which stays the same at n log n. Since some numbers were already in place during quicksort, the overall run time was closer to (n^2). <u>So Merge sort is more efficient in this case.</u>

Question 7 and 8 code are submitted separately