1. Sorting Question
    a. Time complexity is O(1)
    b. The time complexity of bubbleSort is O(n^2) because I used two for loops
    c. The time complexity of selectionSort is O(n^2) because I used two for loops

Part E

Table and Graph for the Quicker sorting algorithms

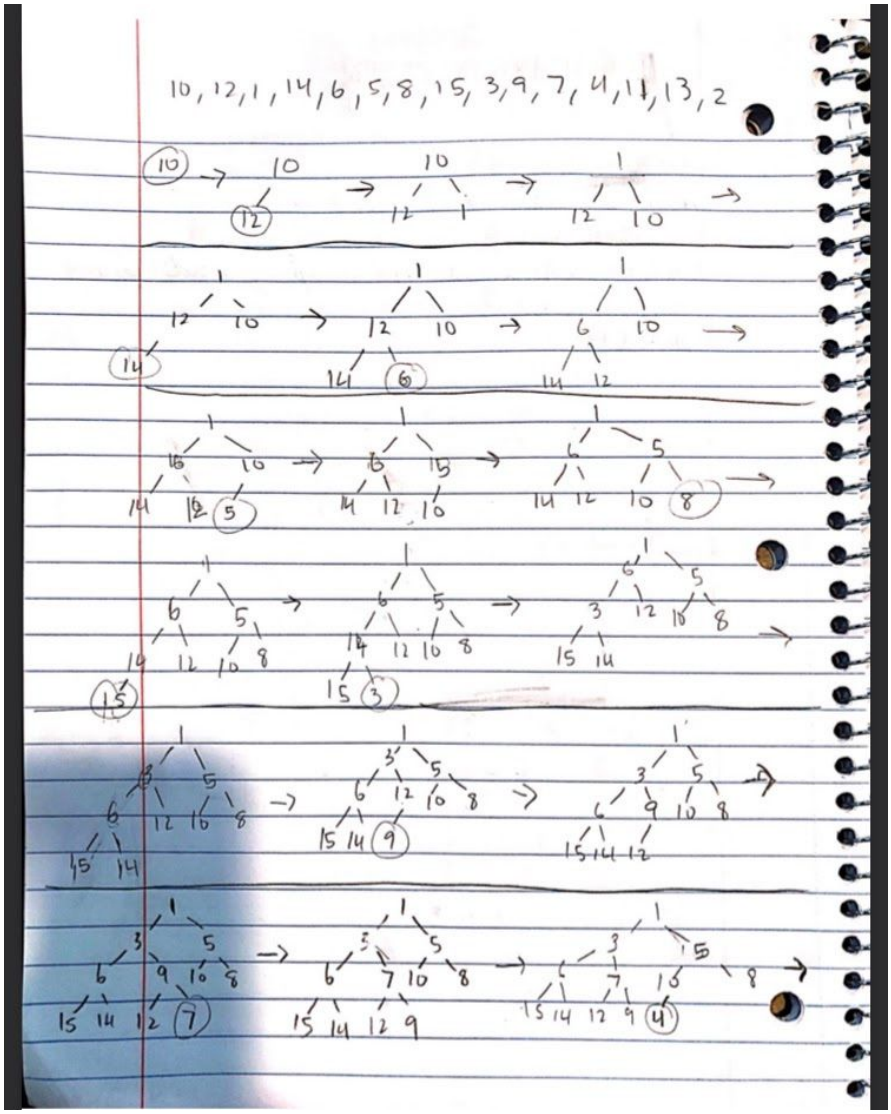
	heap	merge	hybrid	quick	shell
1000	0.000552	0.000831	0.142282	0.000101	0.000462
5000	0.00152	0.001304	0.14734	0.000661	0.000952
10000	0.007134	0.007321	0.149545	0.003479	0.008664
50000	0.015734	0.015988	0.526206	0.008507	0.021937
100000	0.09259	0.090034	0.195061	0.048291	0.125725
500000	0.196238	0.182626	0.612124	0.095541	0.300867
1000000	1.19412	1.07103	0.741575	0.560184	1.9766
2000000	2.48415	2.3127	2.36699	1.19417	4.63504
4000000	4.14341	4.30668	4.02196	2.49216	9.99087
6000000	8.44153	7.37779	5.50329	3.79399	16.2957
8000000	12.1006	9.93383	6.95739	5.20154	23.0986
10000000	15.6457	12.7414	8.45929	6.51641	28.8701
50000000	93.7044	68.9413	38.7917	35.355	187.024
100000000	211.408	138.276	79.1399	74.1599	435.556

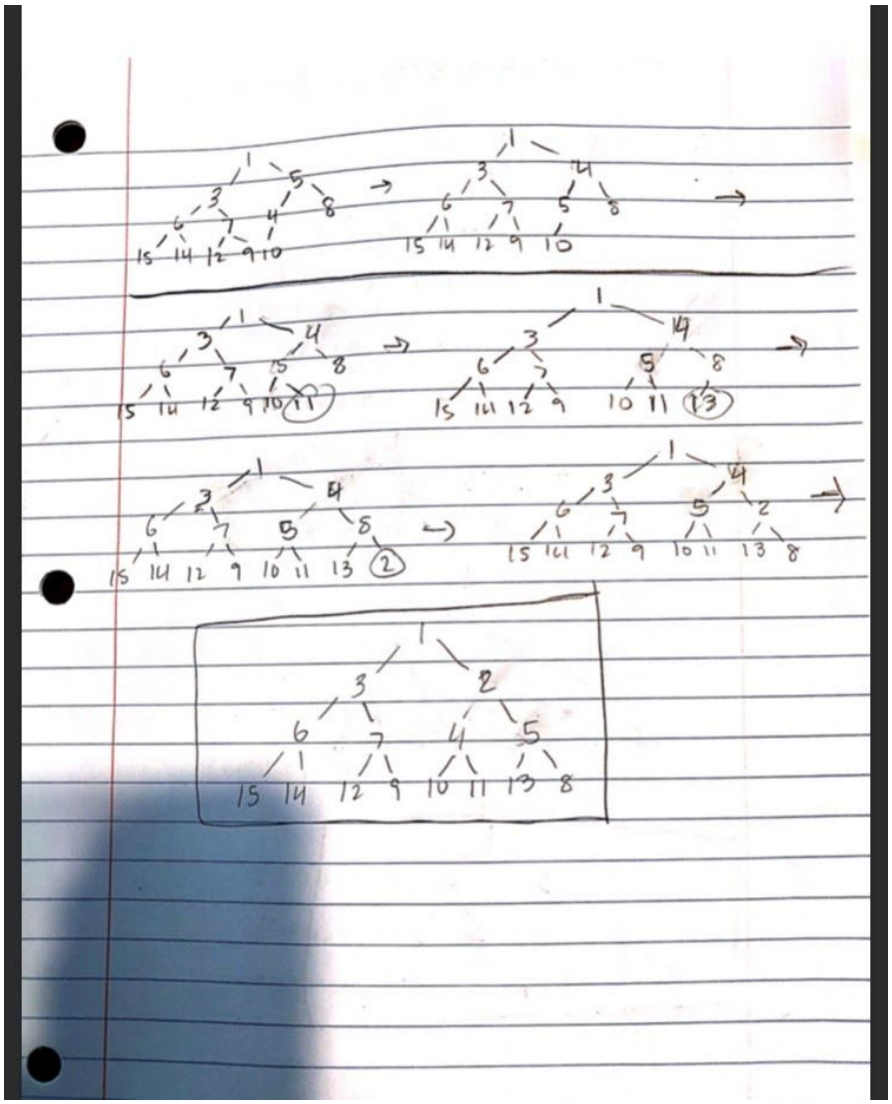
Table and Graph for the slower sorting algorithms

	bubble	selection	insertion
1000	0.024127	0.005314	0.004443
2000	0.090797	0.020694	0.016769
4000	0.37657	0.080613	0.074949
6000	0.84988	0.181541	0.166997
8000	1.50187	0.335637	0.295929
10000	2.37792	0.50569	0.482913
20000	9.36966	1.99933	1.85845

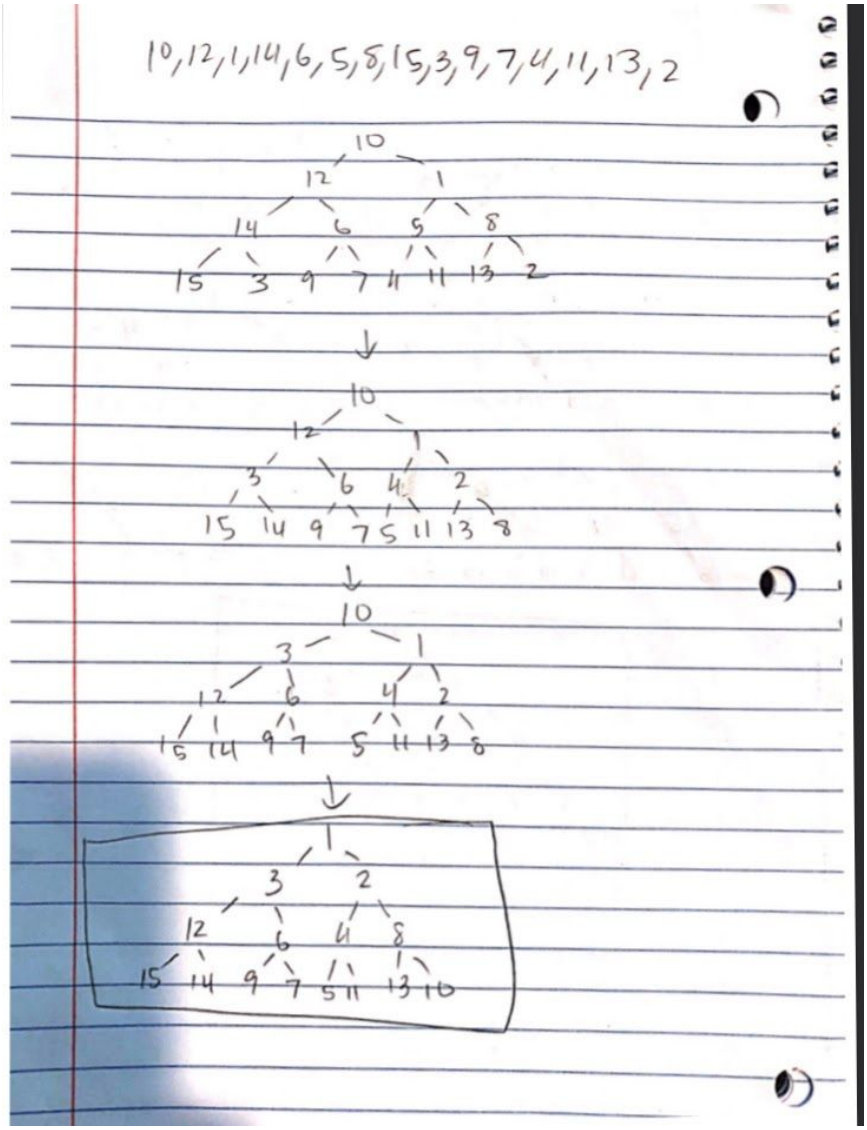
40000	37.767	8.02213	7.45796
60000	85.5619	17.9659	15.782
80000	152.298	31.4049	27.7449
100000	223.105	47.8364	45.9804

Question 2b:
Part A:


10, 12, 1, 14, 6, 5, 8, 15, 3, 9, 7, 4, 11, 13, 2



Part B Linear Time:



Question 2c:
In order to make this function less than O(nlogn), I used the min-heap to build the heap using a linear time algorithm. So the time complexity for that was O(n) and the time complexity for deleting the node was o(klogn).

*There is a file called test.cpp in the PA4 folder. I used this to test certain functions for part one.