LAB 12

Haru Chu, Nicholas Krouse, Jack Vo

The program was programmed and executed via Replit.

Sort Algorithms	Test Size					
	10	100	500	5000	25000	100000
Bubble Sort	0 ms	0 ms	0 ms	58 ms	1672 ms	24355 ms
Insertion Sort	0 ms	0 ms	0 ms	9 ms	238 ms	3880 ms
Merge Sort	0 ms	0 ms	0 ms	1 ms	8 ms	34 ms
Quick Sort	0 ms	0 ms	0 ms	0 ms	2 ms	9 ms
Heap Sort	0 ms	0 ms	0 ms	0 ms	3 ms	17 ms
Counting Sort	0 ms	0 ms	0 ms	0 ms	0 ms	0 ms
Radix Sort	0 ms	0 ms	0 ms	0 ms	2 ms	9 ms

All the sort methods worked well in the array size of less than 5000 with the speed being 0ms in most of the cases aside for Bubble Sort, Insertion Sort, and Merge Sort (58 ms, 9 ms, and 1 ms in the array size of 5000 respectively). When the size started getting bigger, Bubble Sort and Insertion Sort visibly began to fall behind in speed. With the array size of 25000, the speed of Bubble Sort and Insertion Sort was each at 1672 ms and 238 ms while that of other methods was all less than 10 ms. When it came to the size of 100000, all sorting methods were slightly slower than before due to the huge size of array, still the slowest ones were still Bubble Sort with the speed of 24355 ms and Insertion Sort with that of 3880 ms. This result was within our expectations because while Bubble Sort and Insertion Sort were the easier ones among the sorting methods, their weaknesses were clearly shown when dealing with huge sizes of the array due to their big O notations generally being O(n^2) and having to iterate through the whole array.

Contribution:

Haru Chu: Task 3 main

Jack Vo: Task 2, Student.h, LinkedList.h, Task 1

Nicholas Krouse: Debug

Instructions:

The test cases for task 2 and task 3 are marked in the main program. Please comment or uncomment each test case accordingly when testing.

Also, for task 2, we manually input each sort function at a time, so if you want to try it yourself, please put new sort function in, current is the quick-sort function.

Grade Distribution:

Same grade for Jack and Haru

Minus 15 percent for Nick.