**Lab 5: Sorting**

In this lab we covered sorting and efficiency. We were required to make 6 different sorting methods. They were bubble sort, insertion sort, merge-sort, quicksort, counting sort, and radix sort. Then, we had to use the chrono library to record the time it takes for each sorting method to sort through a list of a certain size. The time values were recorded in a table for an array size of 10, 100, 500, 25,000, and 100,000 values. Then, we were required to use 3 of the 6 sorting methods to sort an unordered linked list. We used bubble sort to sort the list using M numbers, insertion sort to sort the list using first names in an alphabetical order, and merge sort to sort the list using last names in an alphabetical order. The coding skills acquired from this lab will help us excel in our future career as computer or software engineers.

**Contributions:**

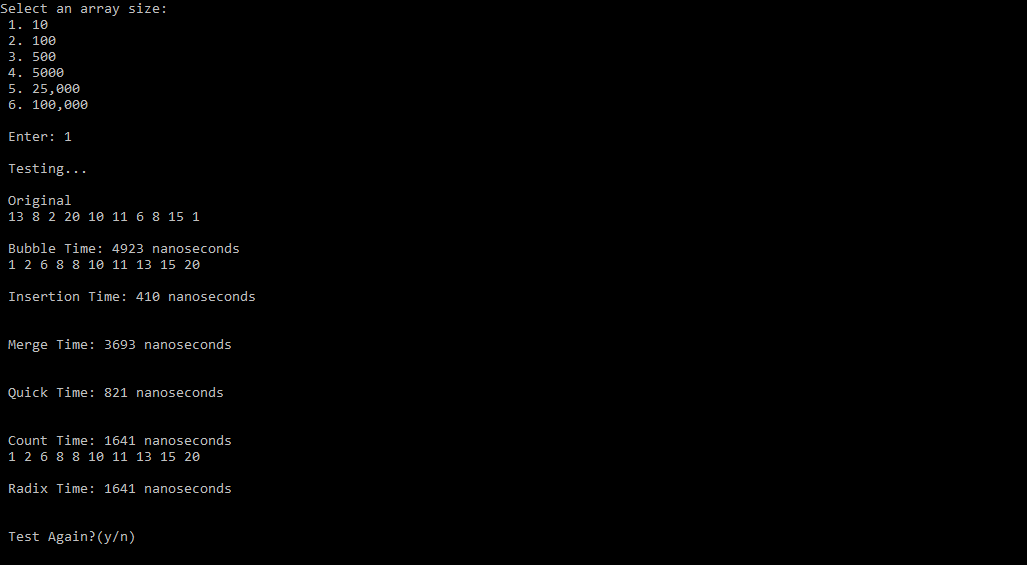
**Raman Hliabovich** – Brainstorming and code revision. Cleaning up mains. Code testing and lab report. Fixed merge sort and insertion sort for linked list.

**Sean Sliter** – Brainstorming and code revisions. Most of Task 3. Made the table for task 2.

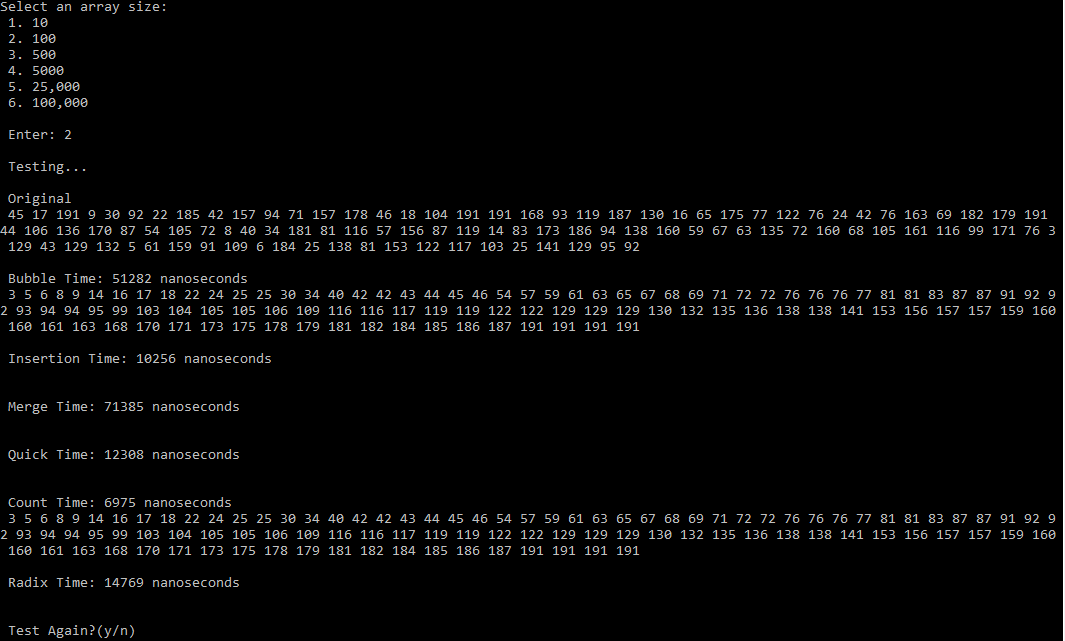
**Zack Johnson** – Brainstorming and code revisions. Most of Task 1. Helped with Task 3.

**Task 1:**

**Task one output screen gives the option to select the array size, prints the original array, and outputs the time it takes for each sorting algorithm to sort the selected array (size 10) in nanoseconds.**



**Output times for a 100 size array.**



**Task 2:**

**Table of times it took to sort the array of different sizes for each sorting algorithm.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **array[10]** | **array[100]** | **array[500]** | **array[5000]** | **array[25000]** |
| **Bubble Sort** | 1763 | 87801 | 2198919 | 287217391 | 5611251159 |
| **Insertion Sort** | 352 | 11637 | 280331 | 18276486 | 417268848 |
| **Merge Sort** | 5289 | 95559 | 662215 | 3315656 | 18267318 |
| **Quick sort** | 1058 | 21509 | 166788 | 1903073 | 10372931 |
| **Counting Sort** | 2116 | 5995 | 34204 | 142810 | 651989 |
| **Radix Sort** | 3526 | 14457 | 118126 | 699592 | 4117860 |

**Graphs for each for each of the sorting algorithms showing the relationship between array size and sorting time.**

Bubble Sort Performance: O(n4)

Insertion sort Performance: O(n4)

Merge Sort Performance: O(n3)

Quick Sort Performance: O(n3)

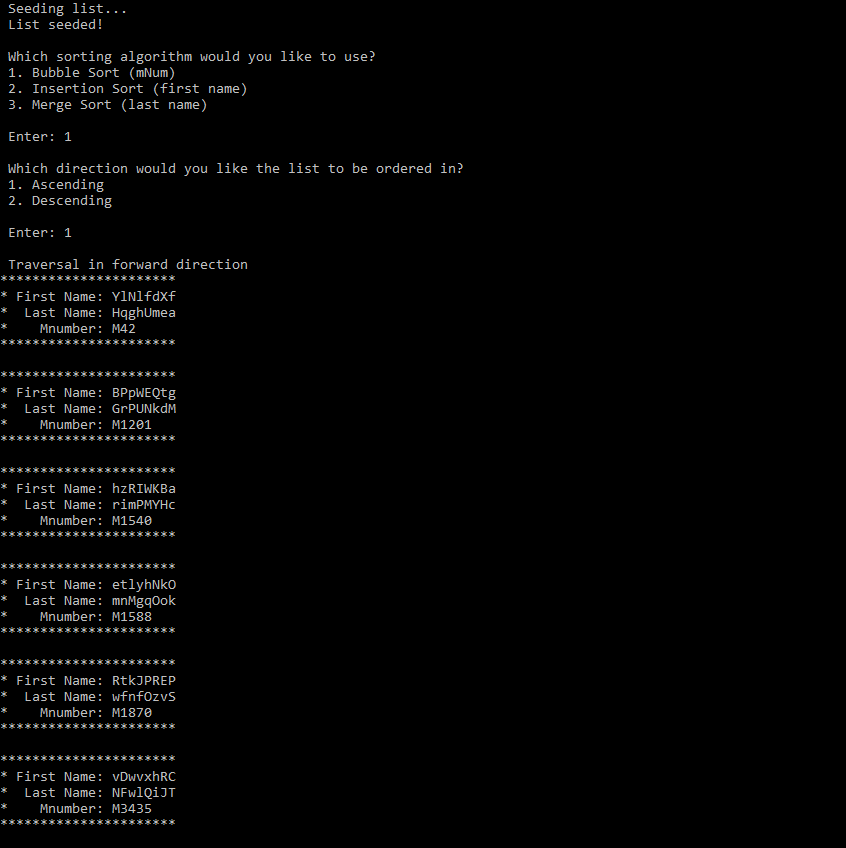
Counting Sort Performance: O(n3)

Counting Sort Performance: O(n4)

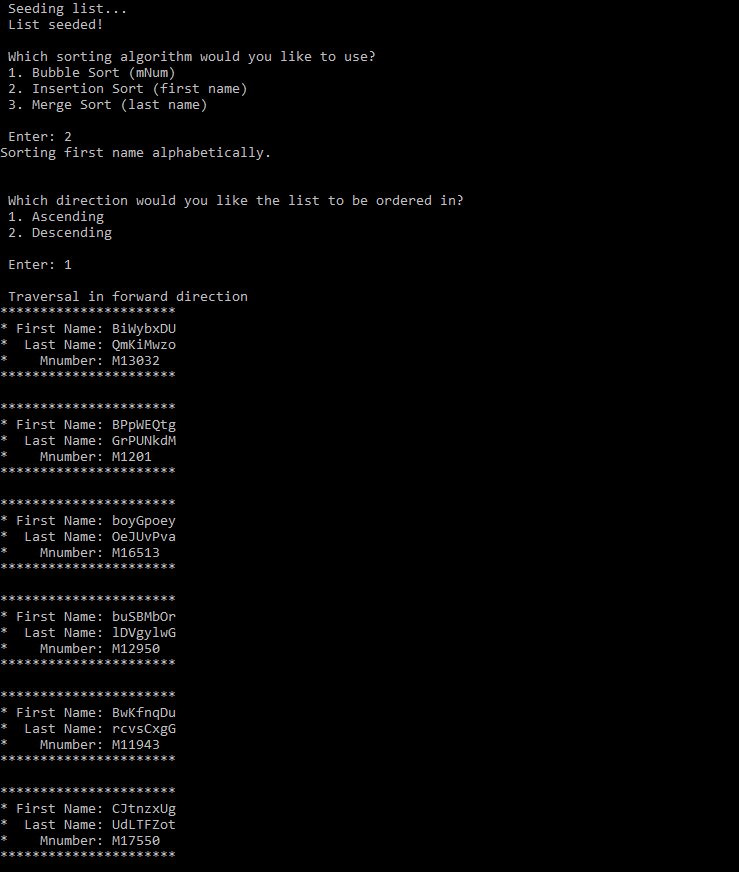
**Task 3:**

**In Task 3 the program generates a linked list of 50 random Mnumbers, first names, and last names. Then it asks you to choose which sorting algorithm you want to use. Next, it asks what direction you want to print the sorted list in and outputs the 50 values in the linked list accordingly.**

**This output shows the shows the sorted linked list that was sorted with a Bubble sort using the Mnumbers to sort the list. Then it outputs the 50 students sorted in the linked list in an ascending direction. (Only the first 6/50 sorted students are shown in the picture because the list is too long to fit in the image below)**



**Sorting list in an alphabetical order with insertion sort using firs names. Printing sorted list in an ascending order.**



**Sorting list in an alphabetical order with merge sort using last names. Printing sorted list in a descending order.**

