lab 13 Fundamental sorting: Quick and Merge Sort

Instructions: Implement Quick and Merge sort for your Array List class. Then implement one of the fundamental $O(nlog_2n)$ sorting algorithm (Quick or Merge) for a (singly or doubly) Linked List.

```

template <class T>
class Array {
    private:
    /* You fill out the private contents. */

    public:
    ...

    /* Runs a quick sort algorithm on the array.
     * The array shall be ordered from least to greatest
     */
    void qsort();

    /* Runs a merge sort algorithm on the array.
     * The array shall be ordered from least to greatest
     */
    void msort();

    /* Runs the sort routing you believe is the best. */
    void sort();
    ...
};


/* SLL = Singly Linked List */
template<class T>
class SLList {
...
    public:
...
        /* Sort the linked list. You may use any O(nlogn) sort algorithm you
         * wish.
         */
        void sort();
...
};
```

Write some test cases:

Create some test cases, using cxxtestgen, that you believe would cover all aspects of your code.

Part 2: Performance
Generate a graph to compare the performance of {bubble sort, selection sort, insertion sort} vs quick sort for an array. Then {bubble sort, selection sort, insertion sort} vs the sort you chose for a Singly Linked List. Your graph should have data size on the x axis and time on the y axis. Make sure to label each graph line! Please turn in as a .pdf!

Memory Management:
Now that are using new, we must ensure that there is a corresponding delete to free the memory. Ensure there are no memory leaks in your code! Please run Valgrind on your tests to ensure no memory leaks!

How to turn in:
Turn in via GitHub. Ensure the file(s) are in your directory and then:

- $ git add <files>
- $ git commit
- $ git push

Due Date: October 06, 2020 2359

Teamwork: No teamwork, your work must be your own.