Sorting Paper

The objective of this assignment is two-fold. First, I want you to get experience implementing a wide-range of sorting algorithms. Second, I want you to get experience with a common task assigned to both new employees and graduate students: basic research.

I want you to assume that you are a new hire at a business of your choice or you are a graduate student researching a topic you find interesting. Your boss or adviser gives you an assignment to help you get your feet wet while you are ramping up. They will need a sorting algorithm for some reason. However, your boss/advisor knows there are a bunch of different sorting algorithms out there all with relative advantages and disadvantages. Your boss wants you to implement a wide-range of different sorting algorithms, conduct experiments using your implementations, and then give he/she a paper that explains which sorting algorithm and why. There is no single correct answer for these assignments. The best algorithm will be entirely dependent on what scenario you are designing for.

The Lab will be due a week from thursday. Oct 29th. The paper will be due Wed Dec 2nd.

You must provide a full testing suite for each algorithm.

If you provide an that your algorithms implement, then you will get 10 bonus points.

First. In the first part, you will implement the following sorting algorithms.

- BubbleSort.
- InsertionSort.
- MergeSort.
- HeapSort.
- QuickSort.
- RandomizedQuickSort.
- RadixSort.
- CountingSort.
- A sort of your choice (not any of the above or BogoSort)

Paper. In the paper you will need to,

- Explain your career/graduate school research.
- Explain why your company/research need a sorting algorithm.
- Explain what requirements come out of the scenario in which you plan to use a sorting algorithm.
- Run experiments to show how the different sorts behave in your scenario. (Remember, your scenario will have a range of possible values. For example, if your scenario worked mostly with very small lists, then you should show how all algorithms worked on lists from size 1 to 1,000. However, if you worked mostly with very large lists, then you probably would want your smallest list to be 100,000 elements. Either way, you want to make sure that your data entirely encapsulates all possible values you would encounter in the real world.)
- The graphs of your data.
- An explanation of what your data shows.
- A choice of which sorting algorithm should be used for your scenario and an explanation of why you chose that algorithm.