1. Implement Shellsort which reverts to insertion sort. (Use the increment sequence 7, 3, 1). Create a table or a plot for the total number of comparisons made in the sorting the data for both cases (insertion sort phase and shell sort phase). Explain why Shellshort is more effective than Insertion sort in this case.

2. The Kendall Tau distance is a variant of the "number of inversions" we discussed in class. It is defined as the number of pairs that are in different order in two permutations. Write an efficient program that computes the Kendall Tau distance in less than quadratic time on average. Plot your results and discuss. Use the dataset provided here. Note: data0.* for convenience is an ordered set of numbers (in powers of two). data1.* are shuffled data sets of sizes (as given by "*").

3. Create a data set of 8192 entries which has in the following order: 1024 repeats of 1, 2048 repeats of 11, 4096 repeats of 111 and 1024 repeats of 1111. Write a sort algorithm that you think will sort this set "most" effectively. Explain why you think so.

4. Implement the two versions of MergeSort that we discussed in class. Create a table or a plot for the total number of comparisons to sort the data (using data set here) for both cases. Explain.

Data Set for Questions above:
https://drive.google.com/file/d/0B4xMi5S-VFVRVWh0YzV6bmFLMjQ/view?usp=sharing

5. Implement Quicksort using median-of-three to determine the partition element. Compare the performance of Quicksort with the Mergesort implementation and dataset from Q4. Is there any noticeable difference when you use N=7 as the cut-off to insertion sort. Experiment if there is any value of "cut-off to insertion" at which the performance inverts.

6. View the following Data Set here. The column on the left is the original input of strings to be sorted or shuffled; the column on the extreme right are the string in sorted order; the other columns are the contents at some intermediate step during one of the 8 algorithms listed below. Match up each algorithm under the corresponding column. Use each algorithm exactly once: (1) Knuth shuffle (2) Selection sort(3) Insertion sort (4) Mergesort(top-down)(5) Mergesort (bottom-up) (6) Quicksort (standard, no shuffle) (7) Quicksort (3-way, no shuffle) (8) Heapsort.

[location of data: https://sakai.rutgers.edu/access/content/group/9a721e60-ef8e-412a-835b-14c0ab9020f0/HW-Dataset/algorithm-stage.png ]