IMPORTANT FIRST STEPS:

1. **Close your laptops and put them away.**
2. **Form a group of 2-3 students.**
3. **Clearly put your names and IDs on 1 copy of this worksheet.**
4. **Be sure to turn this exercise in at the end of class or at the start of next class (check with the instructor). Yes, you turn this one in.**

Sorting

Make sure that you have 10 different playing cards in front of you. It's OK if you have duplicates!

Shuffle the cards and lay them out in front of you, face up, in random order.

Forgetting, for a moment, the sorting algorithms we have talked about in class, select a person to sort the cards as he/she normally would.

Once the cards are sorted, describe this algorithm in pseudocode below:

Does this match any of the algorithms we have done in class?

Is this an efficient algorithm? Why or why not?

What considerations are there for ‘sorting by hand’? (Would these change if you were sorting exam papers, for example?) Are these different than the considerations for sorting digitally?

Next let’s practice some of the other algorithms we have learned.

Insertion Sort:

Again, randomize the cards and deal them out in front of you. Predict how many moves the algorithm should take in the worst, average, and best cases and write those predictions here:

Apply the insertion sort algorithm. First apply the general algorithm, then try the C++ algorithm, pretending that your list of cards in front of you is an array.

In each case, have someone count the steps and record that number here:

How did the number of steps compare to your predictions? Account for any differences and explain.

Mergesort:

Again, randomize the cards and deal them out in front of you. Predict how many moves the algorithm should take in the worst, average, and best cases and write those predictions here:

Apply the mergesort sort algorithm. Apply the general algorithm, pretending that your list of cards in front of you is an array. Use a spare piece of paper to “copy” your array.

Pay careful attention to how you merge-- you need to understand this!

Have someone count the steps and record that number here:

How did the number of steps compare to your predictions? Account for any differences and explain.

Quicksort:

Again, randomize the cards and deal them out in front of you. Predict how many moves the algorithm should take in the worst, average, and best cases and write those predictions here:

Apply the quicksort algorithm. First apply the general algorithm, then try the C++ algorithm, pretending that your list of cards in front of you is an array. Use paper to keep track of your variables when applying the Bentley algorithm. Remember the Bentley algorithm is just a specific implementation-- it is not the abstract algorithm!

In each case, have someone count the steps and record that number here:

How did the number of steps compare to your predictions? Account for any differences and explain.