These must be completed and shown to your lab TA either by the end of this lab, or at the start of your next lab. You may work in groups of up to two people.

1. Download `qsortcount.cpp` from the course web page.
   This implementation of Quicksort is somewhat different from the partition method described in class. Try performing on paper some iterations on small arrays (~8-12 elements) to understand the operation of this implementation.

   The remaining questions are about how to instrument Quicksort in order to evaluate its average case performance empirically.

2. Add the global integer variable `comps` to the program and add one line to `quicksort` to count the number of comparisons between array elements that `quicksort` performs.

   Run your program using array size 1000 for 100 repetitions. What is the average number of comparisons?

   Run your program with a few different choices of `NN` (array size). How does the number of comparisons change with `NN`? Does it match your intuition/knowledge about the average case running time of Quicksort?

   Note: Write code in `main` to do this experiment.

3. You do not really need to sort the array in order to calculate comps for an input array of size n.

   Write a new recursive function `qc` that takes a single parameter `n` and returns the number of comparisons `quicksort` would perform in sorting an array of size n, but does not actually sort anything. (You will call `randint` to pick a random pivot location.)

   Be sure to show your work to your TA before you leave, or at the start of the next lab, or you will not receive credit for the lab!