Homework 3: Sorting (due Tuesday, August 9th at 11:59 PM)

The file sorts.cpp contains an almost complete program that creates a randomly ordered array, sorts it in a few ways, and reports on the elapsed times. Your job is to complete it and experiment with it.

You can run the program as is to get some results for the STL sort algorithm. The insertion sort will give you an assertion violation, because the insertion sort function right now doesn't do anything. That's one thing for you to write.

The objects in the array are not cheap to copy, which makes a sort that does a lot of moving objects around expensive. Your other task will be to create a vector of pointers to the objects, sort the pointers using the same criterion as was used to sort the objects, and then make one pass through the vector to put the objects in the proper order.

Therefore, your two tasks are:

- Implement the insertion_sort function.
- Implement the compareSensorPtr function and the code in main to create and sort the array of pointers.

The places to make modifications are indicated by TODO: comments. You should not have to make modifications anywhere else. (Our solution doesn't.)

Try the program with about 10000 items. Depending on the speed of your processor, this number may be too large or small to get meaningful timings in a reasonable amount of time. Experiment. Once you get insertion sort working, observe the $O(N^2)$ behavior by sorting, say, 10000, 20000, and 40000 items.

To further observe the performance behavior of the STL sort algorithm, try sorting, say, 100000, 200000, and 400000 items, or even ten times as many. Since this would make the insertion sort tests take a long time, skip them by setting the TEST_INSERTION_SORT constant at the top of sorts.cpp to false.

Notice that if you run the program more than once, you may not get the same timings each time. This is not because you're not getting the same sequence of random numbers each time; you are. Instead, factors like caching by the operating system are the cause.

Turn It In

You will use BruinLearn to turn in this homework. Turn in one zip file that contains your solutions to the homework problem. The zip file itself must be named in the following format (no spaces): LastNameFirstName_SID_AssignmentTypeAssignmentNumber.zip (AssignmentType: P=project, H=homework; AssignmentNumber = {1,2,3,4}). An example is BruinJoe_123456789_H3. The zip file must contain only the file `sorts.cpp`, a C++ source file with your solution to the problem.