CS3D5A - Assignment 2 - Sorting. Anton Gerdelan and Peter Lavin
Due: Tue 15th November, midnight. (9 days) **Weighting**: 1/4 of CA grade. 10% of course total.

Deliverables
1. C or C++ source code containing your implementations of sorting algorithms.
2. A (2~3 pages) report with a brief overview of each implementation, and a table of test results.

Instructions

- Implement any sorting algorithm of your choice and also implement quicksort.

- Comment your code to demonstrate you understand the sorting process. Summarise each sort in your report to show your understanding of each algorithm and how this relates to your code.

- Create an array of test data to run your sorting functions on. You may manually assign array values, or generate them randomly. e.g. an array of integers.

- Count the number of investigated elements or probes made by each of the search functions on the test data.

- Put the results in a table comparing your implementations, giving your test results, and the big-O complexities. State the size of the test array used, and if it was randomly distributed, mostly sorted, or sorted but in reverse.

Test Results

Algorithm Name	Probe Count	Big-O Complexity		
		Best	Average	Worst

- Any sort of chart or animation showing one of the sorting processes would add extra value to your report. See any Wikipedia article on any sort for an animation example, or our first slide on sorting for typical chart examples. Using your own test data would be very neat.

Grading Scheme
Code: correct, well commented sorts *(max. 6 points).*
Report: summaries and table *(max. 3 points)* chart *(max. 1 point)*

Suggested Approach
Aim to implement a function for an elementary sort function in the lab, and call it from `main()`. Print the array of test data before and after to ensure that it worked. Use a global counter for each function to add the number of probes and print this too. Write a few lines describing the sort and add a row to your table. If you are finding this slow progress - ask questions. Consider submitting at end of day 1, leaving time to upgrade with quicksort and resubmit. You will need to do some reading and on-paper step-throughs to understand quicksort - schedule time for this over reading week.

Considerations
- Do not expect deadline extensions - we don't have spare time left. We will monitor the discussion boards over reading week if you need technical help or advice.
- You may want to increase the size of your test array to generate interesting results. If it is very large it will be harder to chart and debug. Start small and easy.
- Remember to use the same original test data to compare both sorts - but *don't give an already sorted array as input to your next sort*!
- Be careful with the *first* and *last* arguments given to each sort - is the "*last"* parameter the last valid index or the size of the array?
- If you feel very confident, you can modify your image drawing code from the warm-up assignment to draw a chart of values in your array or create a .gif animation of a sort in action.