Assignment 6

Notes:

- Include counts of compares and swaps in **all** trace tables **for each row**, and a **total** count when done.
- Do all trace tables by hand (don't write programs to generate them – you will learn more).

Question 1

Do a trace table for **selection** sort for the following array of numbers: **15 12 16 13 11 14 10 17**.

Do a trace table for **selection** sort for the following array of numbers: **10 11 12 13 14 15 16 17**.
What does this say about **best** case performance of **selection** sort?

Do a trace table for **selection** sort for the following array of numbers: **17 16 15 14 13 12 11 10**.
What does this say about **worst** case performance of **selection** sort?

Question 2

Do a trace table for **insertion** sort for the following array of numbers: **15 12 16 13 11 14 10 17**.

Do a trace table for **insertion** sort for the following array of numbers: **10 11 12 13 14 15 16 17**.
What does this say about **best** case performance of insertion sort?

Do a trace table for **insertion** sort for the following array of numbers: **17 16 15 14 13 12 11 10**.
What does this say about **worst** case performance of **insertion** sort?

Question 3

In this question you will measure the performance of selection sort and insertion sort by plotting **N** against the average number of operations (comparisons plus swaps of array elements) needed to sort a randomly generated array of **N doubles**.

Write (static) methods:

- **boolean isSorted(double[] a)** – returns true if and only if the array **a** is sorted.
- **double[] random(int N)** – returns an array of doubles of size **N** where each entry in the array is a random double between 0 and 1 (**use Math.random()**).
- **void selection(double[] a)** – sorts the array **a** using selection sort.
- **void insertion(double[] a)** – sorts the array **a** using insertion sort.

Instrument your two sorting routines to count the number of operations (compares plus swaps of array elements) each routine does.

Write a main method which, for **various appropriate** choices of N:

- generates 100 random arrays of size **N**, calls **selection** sort on each, checks that each array is sorted, and outputs the average number of operations per call.
- generates 100 random arrays of size **N**, calls **insertion** sort on each, checks that each array is sorted, and outputs the average number of operations per call.

Do a plot (in Excel, OpenOffice, etc) of N against the number of operations (plot both sorts on the same plot), labeling the axis nicely, etc. Also plot $\mathbf{N^2}$, $\mathbf{\frac{1}{2}\,N^2}$ and $\mathbf{\frac{1}{4}\,N^2}$, all on the same axis.

Remark:

- Be careful not to sort on an already sorted array (doing so means you are measuring the worst/best case, and not the average case).
- Strictly speaking you don't need to generate 100 arrays and average for **selection** sort since its best case, worst case and average case all the same.
- Make sure your sorting and instrumentation code is correct before you start measuring the counts and doing the plots!