1 Assignment 1

Submit source code and running instructions to EAS[1]. Submit the textual components in a single text/pdf file, providing the assignment numbering to help your marker. Do the code in Java. Do not use Java's search or sort methods! Place textual responses for 2 and 3 in a separate text/pdf file!

This assignment should use the Command-Line, reading the String[] provided to *main*. **Do not use Scanner, do not use anything else**. Command-Line input is what you must use. Learn about String[] args in the main method!

Posted: Monday, May 7^{th}

Due: Tuesday, May 15^{th}

Grade: 5%

1. Given an array of unsorted integers, for example *99 37 17 5 12 33*, write a program that performs an Insertion Sort and outputs the results. This program should be in $\Theta\left(n^2\right)$ for time in the average case (for the version without "debug").

   (a) Make sure your program allows as input on the command-line a space-separated list of integers (the above being just an example).

      i. The file name should be ISort.java

      ii. format: java ISort [<debug>] <vals...>

      iii. e.g.: java ISort 99 37 17 5 12 33

      iv. e.g.: java ISort debug 99 37 17 5 12 33

   (b) When the program completes, it should display a line with the sorted values, and indicate how many nanoseconds it took to sort the values.

   (c) When debug is passed as the first argument, print the current array to standard out after each compare, with the current value being surrounded by square brackets and the value being looked at surrounded by 'i's; at the end of the line print a space and then a '<', '>' or '=', depending on how the current values relates to the value being looked at. If the value is at the beginning of the list, print the values with a '|' at the end of the line. When in "debug" mode, don't worry about the impact on the the complexity of doing the output.

Insertion Sort Example

%java ISort debug 99 37 17 5 12 33

[99] 37 17 5 12 33 |	17 i37i 99 [5] 12 33 <	5 12 17 37 i99i [33] <
i99i [37] 17 5 12 33 <	i17i 37 99 [5] 12 33 <	5 12 17 i37i 99 [33] <
99 [37] 17 5 12 33 |	17 37 99 [5] 12 33 |	5 12 i17i 37 99 [33] >
37 i99i [17] 5 12 33 <	5 17 37 i99i [12] 33 <	5 12 17 33 37 99
i37i 99 [17] 5 12 33 <	5 17 i37i 99 [12] 33 <	completed in 1291555ns
37 99 [17] 5 12 33 |	5 i17i 37 99 [12] 33 <	
17 37 i99i [5] 12 33 <	i5i 17 37 99 [12] 33 >	

!!You don't have to implement columns! I'm just saving space here.

2. In clear, natural language, describe the average runtime of this program after you have empirically tested it with random data up to 1000 values (don't use debug or your times will be really bad). Calculate an operation cost based on this. Show your data table!

   (a) This textual response should be no more than 5 lines / 50 words.

   (b) Include as your data table the length of the array, the times run and the average time in ns (on each line)

[1] https://fis.encs.concordia.ca/eas/

3. In clear, natural language, describe how your algorithm's complexity would change if you changed the algorithm to use a binary search instead of searching previous values until you found where to insert your current value. Describe when this would be useful.

   (a) This textual response should be no more than 5 lines / 50 words.