COSC 311 WINTER 2010 Project #3 Using a Heap

Distributed: March 18, 2010 **Due**: April 6, 2010
Modified to make more sense of the second sweep: March 31, 2010

Brief description:
Out of a set of ~~10,000~~ 100,000 random `int` values, find the 10 smallest values using a heap (duplicated 10-smallest values are OK). Also output the following:

(1) maximum (`max`) value in the data set,
(2) minimum (`min`) value in the data set,
(3) the value `(min + max)/2,` which we will call the average (average is a double). See `http://wikipedia.org/Average` for various kinds of averages (not including this one).
(4) the count of the number of items that are less than `(min + max)/2`
(5) the count of the number of items equal to `(min + max) /2`
(5) the count of the number of items that are greater than `(min + max)/2`
(6) elapsed time for entire program execution (in milliseconds or microseconds).

More specification:
With a separate program, generate a data file containing ~~10,000~~ 100,000 random `int` (not `long`!) values; the data values can be duplicated in the data file. The data file you generate can be stream input or record input to the 10-smallest program.

You will sweep through all the data twice.

- During the first sweep, use the heap code to find the 10 smallest values. Also during the first sweep, find the min and the max values. As soon as the first sweep is completed, you can compute the average, `(min + max)/2`.
- During the second sweep, use the average to keep three counts:
  - number of data element less than the average,
  - number of data elements equal to the average,
  - number of data elements greater than the average.

Since the average is a double, you cannot test with '=='! Instead use a smallish ε = `0.03125`. (ε = 1/32)

- The test for less than average is `x < average -ε.`
- The tests for equal to average is `average - ε <= x <= average + ε.`
- The test for greater than average is `average + ε < x.`

If you're using a record file, you won't have to close then reopen the file.

~~Maximum and minimum values in the data set must be determined as a part of the 10-smallest program by a method called on the data file AFTER completing the heap-related code: i.e., completely sweep the file a second time, keeping track of current max and current min values.~~

You cannot use the Priority Queue API in the Java library in your final code: you have to implement a priority queue. Your implementation must be either 1D array or binary tree. *You may cannibalize priority queue code from the web or any text, but your source must be properly credited in your listing.* You are welcome to use the Java library's Priority Queue API during the development phase.

Comment:
You are permitted to use a heap of any size, including size 100,000 or size 10. You should use additional variables as needed. Note the comment in the **Grading** section.

Turn in:
1. Program code to generate data.
2. Program code to compute 10-smallest, max, min, etc.
3. Nicely formatted output of second program (10-smallest, etc).
4. Upload your datafile and your java code (two programs) to your web space at `people.emich.edu`, and turn in three URLs.
5. Turn in screen shot showing results.

Grade based on:

Meeting specifications:	80%
Code standards:	10%
Elegance:	10%

Extra credit: additional 10% is given to the 9 projects with the smallest elapsed time (10-smallest) that satisfy the specs completely.