COS 126 **General Computer Science** **Spring 2016**

Programming Exam 2

Instructions. This exam has one question. You have 50 minutes. The exam is *open course materials*, which includes the course textbook, the companion booksite, the course website, your course notes, and code you wrote for the course. Accessing other information or communicating with a non-staff member (such as via email, instant messenger, text message, Facebook, phone, or Snapchat) is prohibited.

Submission. Submit your solution electronically, via the link on the *Class Meetings* page. Be sure to click the *Check All Submitted Files* button to verify your submission.

Grading. Your program will be graded for correctness, clarity (including comments), design, and efficiency. You will receive partial credit for a program that correctly implements some of the required functionality. You will receive a substantial penalty if your program does not compile or adhere to the prescribed API.

Discussing this exam. Discussing or communicating the contents of this exam before solutions have been posted is a violation of the Honor Code.

This exam. You must turn in this exam. Print your name, NetID, and precept in the space below. Write and sign the Honor Code pledge.

Name:

NetID:

Precept:

"I pledge my honor that I have not violated the Honor Code during this examination."

____

____

____

____

Problem. Create an abstract data type `RollingStats` that supports adding data values to a data structure (one at a time), and computing the *mean* and *k-rolling mean* of those values.

- The *mean* is the average of all data values added to the data structure.
- The *k-rolling mean* is the average of the last k data values added to the data structure; if there are fewer than k data values, it is the average of all data values.

Step-by-step calculation (for reference). This table shows the mean and k-rolling mean immediately after adding each data value x_i, for $k = 3$ and $k = 4$.

			rolling mean	
i	x_i	*mean*	$k = 3$	$k = 4$
1	15	15 / 1	15 / 1	15 / 1
2	16	31 / 2	31 / 2	31 / 2
3	13	44 / 3	44 / 3	44 / 3
4	12	56 / 4	41 / 3	56 / 4
5	14	70 / 5	39 / 3	55 / 4
6	11	81 / 6	37 / 3	50 / 4

(16 + 13 + 12 + 14) / 4

(13 + 12 + 14) / 3

5th data value added

(15 + 16 + 13 + 12 + 14) / 5

API specification. Your program `RollingStats.java` must be organized as an abstract data type with the following API:

`public class RollingStats`	
`public RollingStats(int k)`	*creates a new object with window length k*
`public void add(double x)`	*adds the data value x to the data structure*
`public double mean()`	*returns the overall mean*
`public double rollingMean()`	*returns the k-rolling mean*
`public static void main(String[] args)`	*unit tests this data type (see facing page)*

Corner cases. You may assume that `k` is a positive integer. If no data value has been added, `mean()` and `rollingMean()` should return `Double.NaN`.

Performance requirements (for full credit). Use space proportional to k. The constructor, `add()`, and `mean()` should take constant time; the method `rollingMean()` should take time proportional to k (or better).

Hint: To implement `rollingMean()`, maintain the last k data values in an appropriately chosen collection type from Chapter 4 (i.e., `Stack`, `Queue`, or `ST`).

Input/output specification. The `main()` method should take a positive integer `k` as a command-line argument; read the data values from standard input; and print to standard output the mean and *k*-rolling mean after reading each data value.

- Assume that standard input consists of a sequence of floating-point numbers, separated by whitespace.
- For each data value, print one line of output that consists of the mean and *k*-rolling mean (with each formatted using 2 digits of precision after the decimal point), separated by whitespace.

Here are two sample executions, corresponding to the example on the facing page:

```
% more input.txt
15.0
16.0
13.0
12.0
14.0
11.0

% java-introcs RollingStats 3 < input.txt
15.00  15.00
15.50  15.50
14.67  14.67
14.00  13.67
14.00  13.00
13.50  12.33

%java-introcs RollingStats 4 < input.txt
15.00  15.00
15.50  15.50
14.67  14.67
14.00  14.00
14.00  13.75
13.50  12.50
```

Test file. For convenience, the file `input.txt` is available at

`http://introcs.cs.princeton.edu/java/input.txt`

Submission. Submit the single file `RollingStats.java` via Dropbox at

`https://dropbox.cs.princeton.edu/COS126_S2016/Exam2`

You may assume access to the standard libraries in `stdlib.jar`, along with the collection types from Chapter 4 (`Stack`, `Queue`, or `ST`).