Lies, damned Lies, and

This problem is worth 12 points. In this problem we will have some fun with the statistics. You are to implement twelve (12) methods in the `LieCalculator` class. The twelve methods are `getMean()`, `getMedian()`, `getMode()`, `getRange()`, `getMidRange()`, `getStandardDeviation()`, `getZscore`, `makeMean()`, `makeMode()`, `makeRange()`, `makeMidRange()`, and `makeStandardDeviation()`.

The `LieCalculator` class has a constructor with an `int[] nums` as its only parameter. The values in `nums` are to be saved in the instance variable `data`. It is important the values in `nums` are never changed by any method in the `LieCalculator` class.

The `getMean()` method returns the mean (a `double`) of the values in the `int[] nums`. According to mathsisfun.com, the mean is the average of the numbers. It is easy to calculate: add up all the numbers, then divide by how many numbers there are. In other words, it is the sum divided by the count.

The following code shows the results of the `getMean()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getMean();`	22.25

The `getMedian()` method returns the median of the values in the `int[] nums`. According to mathisfun.com, the median is the "middle" of a sorted list of numbers. To find the Median, place the numbers in value order and find the middle number. Example: the Median of {10, 2, 38, 23, 38, 23, 21, 23}.

Put them in order: {2, 10, 21, 23, 23, 23, 38, 38}
The middle number is 23, so the median is 23.

The following code shows the results of the `getMedian()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getMedian();`	23

When there are two middle numbers we average them. Example: the Median of {10, 11, 15, 15, 13, 16, 23, 10}.

Put them in order: {10, 10, 11, 13, 15, 15, 16, 23}
The middle numbers are 13 and 15, so the median is (13+15) / 2 = 14.

The following code shows the results of the `getMedian()` method.

The following code	Returns

`int[] nums = new int[] {10, 11, 15, 15, 13, 16, 23, 10};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getMedian();`	14

Lies, damned Lies, and

The getMode() method return a sorted int[] containing the median(s) of the values in the int[] nums. According to VirtualNerd.com, The mode of a data set is the number that occurs most frequently in the set. For example, the Mode of {10, 2, 38, 23, 38, 23, 21, 23} is 23 as it is the value that occurs most frequently.

The following code shows the results of the getMode() method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`Lc.getMode().length;`	1
`lc.getMode()[0];`	23

If two or more values occur most frequently, then getMode() should return a sorted int[] containing all values that occurred most frequently. (The values must be sorted in from smallest to largest.) For example, the Mode of {10, 11, 15, 15, 13, 16, 23, 10} is 10 and 15 as those values are the values that occur most frequently.
Note: If no value occurs more than once, return null.

The following code shows the results of the getMode() method when the .

The following code	Returns

`int[] nums = new int[] {10, 11, 15, 15, 13, 16, 23, 10};`
`LieCalculator lc = new LieCalculator(nums);`	

`Lc.getMode().length;`	2
`lc.getMode()[0];`	10
`lc.getMode()[1];`	15

The getRange() method returns the range of the values in the int[] nums. According to mathisfun.com, the range of a data set is the difference between the lowest and highest values.

The following code shows the results of the getRange() method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getRange();`	36

Lies, damned Lies, and

The `getMidRange()` method returns the midrange of the values in the `int[] nums`. According to http://www.montereyinstitute.org, to find the midrange, add together the least and greatest values and divide by two, or in other words, find the mean of the least and greatest values.

The following code shows the results of the `getMidRange()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getMidRange();`	20 = ((2+38)/2

The `getStandardDeviation()` method returns the standard deviation of the values in the `int[] nums`. According to mathisfun.org, The Standard Deviation is a measure of how spread out are the numbers.

OK. Let us explain it step by step.

Say we have a bunch of numbers like 10, 2, 38, 23, 38, 23, 21, 23

To calculate the standard deviation of those numbers:

1. Find the Mean :

   (10 + 2 + 38 + 23 + 38 + 23 + 21 + 23) / 8 = 22.25
2. Sum all of the following: For each number:

   subtract the Mean and square the result:

   $(10 – 22.25)^2 + (2 -22.25)^2 + (38 – 22.25)^2 +$

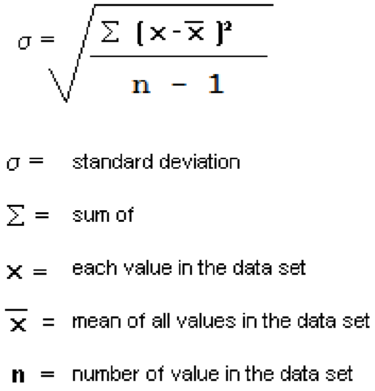
   $(23 -22.25)^2 + (38 – 22.25)^2 + (23 -22.25)^2 + (21 -22.25)^2$

   $+ (23 – 22.25)^2 = 1059.5.$
3. Divide the sum from step 2 by one less than the

   number of values: 1059.5 / 7 = 151.3571429
4. Take the square root of the value from step 3: $\sqrt{151.3571429}$ = 12.30272908

You are done!

In symbols



$$\sigma = \sqrt{\frac{\sum (x-\bar{x})^2}{n - 1}}$$

σ =	standard deviation
$\sum$ =	sum of
x =	each value in the data set
$\bar{x}$ =	mean of all values in the data set
n =	number of value in the data set

The following code shows the results of the `getStandardDeviation()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getStandardDeviation();`	12.3027

Lies, damned Lies, and

The `getZscore(int num)` method returns the z-score of the parameter `num` with respect to the values in `int[] nums`. According to https://www.statisticshowto.datasciencecentral.com/probability-and-statistics/z-score/, the z-score of a value `num` is the number of standard deviations below (a negative z-score) or above (a positive z-score) the mean the value `num` is.

In Symbols, the z score formula for a value x_i with mean $\bar{x}$ and standard deviation s is given by: $z = \frac{x_i - \bar{x}}{s}$

The following code shows the results of the `getZscore()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.getZscore(20);`	-0.182886 = (20 - 22.25) / 12.3027

The `makeMean(int newMean)` method returns the value that is needed to be added to the `int[] nums` which would change the mean to the parameter value `newMean`.
Note: you may assume there exist an `int` that when added to `nums` will change the mean to `newMean`.

The following code shows the results of the `makeMean()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.makeMean(23);`	29

The `makeMode(int newMode)` method returns the number of times `newMode` must be added to the `int[] nums` to make `newMode` the only mode.

The following code shows the results of the `makeMode()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23};`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.makeMode(11);`	4
`lc.makeMode(38);`	2

Lies, damned Lies, and

The `makeRange(int newRange)` method returns an `int[]` containing the two values that when each is independently added to `int[] nums` the range of the new array is equal to `newRange`. The value at index 0 is the smaller value and the value at index 1 is the larger value. To clarify, when the value at index 0 is added to `int[] nums` the range of the new array is equal to `newRange`. And when the value at index 1 is added to `int[] nums` the range of the new array is equal to `newRange`.
You may assume `newRange > getRange()`

The following code shows the results of the `makeRange()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23}`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.makeRange(40)[0];`	-2
`lc.makeRange(40)[1];`	42

The `makeMidRange(int newMidRange)` method returns the value that when added to `int[] nums` would change the midrange to `newMidRange`.
You may assume `newMidRange != getMidRange()`

The following code shows the results of the `makeMidRange()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23}`
`LieCalculator lc = new LieCalculator(nums);`	

`lc.makeMidRange(22);`	42
`lc.makeMidRange(18);`	-2

The `makeStandardDeviation(double newSD)` method returns the value that when added to `int[] nums` would change the standard deviation to `newSD`. You may assume there exist at least one `int` such that adding it to `int nums[]` would change the standard deviation to `newSD`. If more than one value exists, return the value closest to zero. If both values are equally close to zero, return the positive value.
Note: you may assume there exist an `int` that when added to `nums` will change the standard deviation to `newSD`.

The following code shows the results of the `makeStandardDeviation()` method.

The following code	Returns

`int[] nums = new int[] {10, 2, 38, 23, 38, 23, 21, 23}`
`LieCalculator lc = new LieCalculator(nums);`	

`lc2.makeStandardDeviation(11.54459951);`	25
`lc2.makeStandardDeviation(12.94003263);`	40