CMSC 113 – COMPUTER SCIENCE 1 (Prof. Kumar)
Lab#8-Function Libraries

In this lab you will build a small library of useful statistics functions for analyzing numerical data stored in an array. This library is similar in functionality to the `StdStats` library defined in your text.

Task#1: Study the functions in the **StdStats** library below (we have deleted the plotting functions). Carefully note the signatures of each function (*name*, *return value*, and *parameters*), and think about how each function will be defined. We will provide some details below.

`public class StdStats`

`double max(double[] a)`	*largest value*
`double min(double[] a)`	*smallest value*
`double mean(double[] a)`	*average*
`double var(double[] a)`	*sample variance*
`double stddev(double[] a)`	*sample standard deviation*
`double median(double[] a)`	*median*

Task#2: To begin, we will first create a program that reads in some data in an array. The data will be input from standard input (i.e. keyboard or by I/O redirection). The following program reads in some data in an array and then prints it out:

```
public class Stats {
   public static void main(String[] args) {
      double[] data;               // The data array

      // Read the data array
      int n = StdIn.readInt();     // Read the size of the data, n
      data = new double[n];        // Create the data array of size, n
      readData(data);

      // output the data array
      printData(data);

   } // main()

   public static void readData(double[] a) {
      for (int i = 0; i < a.length; i++)
         a[i] = StdIn.readDouble();
   } // readData()
} // Stats
```

Study the program above carefully and answer the questions below:

1. What is the size of the **`data`** array? _____

2. How does **`readData()`** know how many items to read? _____

Notice that, in the program above, the definition of **`printData()`** function is missing. Write the definition of the function **`printData()`** below:

Next, in a file **`Stats.java`** enter the complete program. Compile and run it. You can use the following data as input:

```
5
3.0
1.0
4.0
5.0
2.0
```

Ensure that the program is able to read, and print the data correctly before proceeding.

Next, create a data file containing the same data as above (call it, data.txt). Run you program using the command (using I/O redirection):

```
$ java-introcs Stats < data.txt
```

You should again obtain the same output.

Task#3: Next, let us start to write the basic statistics function defined above. One at a time, define the functions **`min()`**, **`max()`** and **`mean()`**. Then, modify the **`main()`** function to use these functions (as shown below):

```
public static void main(String[] args) {
      double[] data;               // The data array

      // Read the data array
      int n = StdIn.readInt();     // Read the size of the data, n
      data = new double[n];        // Create the data array of size, n
      readData(data);

      // output the data array
      // printData(data);

      // Do some stats on the data
      StdOut.printf("      min %7.3f\n", min(data));
      StdOut.printf("     mean %7.3f\n", mean(data));
      StdOut.printf("      max %7.3f\n", max(data));

   } // main()
```

Before proceeding, ensure that the program is producing correct results.

Task#4: Let us test your program on some real data. Download the AAPL.txt file in from your class web page (next to where this lab-Lab8 was posted). It contains 209 data items[i]. These are the closing stock prices for Apple Inc. for each trading day from January 1, 2020 to October 27, 2020.

Run your program on this data:

```
$ java-introcs Stats < AAPL.txt
```

The output should be as shown below:

```
   min    55.840
  mean    89.568
   max   134.180
```

That is, in 2018, Apple Inc.'s stock traded as low as $55.84 and as high as $134.18 with an average price of $89.568. You can also confirm these results by running the book's `StdStats` program on this data:

```
$ java-introcs StdStats < AAPL.txt
```

It will print out other statistics that we have not yet implemented. Let us do that next.

Task#5: Implement the function **`var()`** to compute the variance of the dataset. Variance is computed using the formula:

$$\sigma^2 = ((a_0 - \mu)^2 + \ (a_1 - \mu)^2 + \cdots + (a_{n-1} - \mu)^2)/(n-1)$$

Where μ is the mean. Write the function **`var()`** below. Then implement and test it. Confirm your results with those from **`StdStats`**.

Task#5: Implement the **`stddev()`** (standard deviation) function. The standard deviation is defined as follows:

$$\sigma = \sqrt{((a_0 - \mu)^2 + \ (a_1 - \mu)^2 + \cdots + (a_{n-1} - \mu)^2)/(n-1)}$$

Once done, test both functions and compare the results with those from **`StdStats`** to ensure their correctness.

`StdStats` gives you all this functionality. But now you also know how it was defined, and in future you can design your own useful libraries similarly.

Task#6: Finally, the three functions defined in **`StdStats`** that we did not implement are used for plotting data:

`public class StdStats`

`double max(double[] a)`	*largest value*
`double min(double[] a)`	*smallest value*
`double mean(double[] a)`	*average*
`double var(double[] a)`	*sample variance*
`double stddev(double[] a)`	*sample standard deviation*
`double median(double[] a)`	*median*
`void plotPoints(double[] a)`	*plot points at (i, a[i])*
`void plotLines(double[] a)`	*plot lines connecting (i, a[i])*
`void plotBars(double[] a)`	*plot bars to points at (i, a[i])*

Modify your **`main()`** function in **`Stats`** as shown below to plot the daily stock prices of Apple Inc. as a bar graph:

```
public static void main(String[] args) {
      double[] data;               // The data array

      // Read the data array
      int n = StdIn.readInt();     // Read the size of the data, n
      data = new double[n];        // Create the data array of size, n
      readData(data);

      // output the data array
      // printData(data);

      // Do some stats on the data
      StdOut.printf("      min %7.3f\n", min(data));
      StdOut.printf("     mean %7.3f\n", mean(data));
      StdOut.printf("      max %7.3f\n", max(data));
      StdOut.printf("      var %7.3f\n", var(data));
      StdOut.printf("   stddev %7.3f\n", stddev(data));

      // Plot the data
      StdDraw.setYscale(min(data)-50, max(data)+50);  // Sets the canvas
      StdStats.plotBars(data);

   } // main()
```

Change the plot command to try **`StdStats.plotLines()`** as well as **`StdStats.plotPoints()`**.

We hope you enjoyed this lab!

Once completed, please save the bar graph from above in a file Lab8.png and put it in the Dropbox folder. Also, as always, send an e-mail to your instructor.

[i] Source: finance.yahoo.com