CISC 3130 Review Assignment HW#1

Write a complete Java program to do the following:

The main program will call a series of methods to process a set of data. One method will read data into an array. A second method will print the values stored in an array in a neat format. Another method will find the average of the values stored in an array. A fourth method will construct a new array from an existing array.

Here are the details on the main program:

1. The main program will read in a parameter value (which the main program will call *size*). Note that the main program reads this value, not one of the methods.

The main program will call a method **readData** to read *size* items into an array (which the main program will call *first* or something else of your choice).

2. The main program will call a method **printArray** which will print the values stored in the first array. Before the main program calls the printing method, it should print a heading saying what is being printed (e.g., the heading could say "here is the original array").

3. The main program will call a method **findAverage** which will find the average of the size values stored in the first array. The main program will print this value, together with a relevant message.

4. The main program will call a method **howFarAway** which will construct a new array called *second* from the original array *first*. (The details are described below.)

5. The main program will call the method **printArray** again, to find the values stored in the new array made in part 4. The main program will print an appropriate message (e.g., “here is the new array”) before it calls this method.

6. The main program will call the method **findAverage** again, to find the average of the values stored in the new array made in part 4. The main program will print the value with a message.

Here are the details on the methods:

A. The method **readData** will read data into an array. The method will receive two parameters: an integer *n*, and an array of doubles called *numbers*. At the beginning, *n* will have a value, and the array will be empty. The method will read *n* lines of data. Each line of data should contain a number of type double (so it might have decimal places). The values stored in this array will be sent back to the main program, to be used throughout the program.

B. The method **printArray** will receive two parameters, an integer *q*, and an array of doubles called *numb*. It will print the *q* values stored in the array *numb*, spaced neatly 5 across a line, with two decimal places for each value. Format the output to create 5 neat columns. For example, if *q* is 13, then 5 values will print on the first line, 5 on the second line,

and 3 on the third, as -- -- -- -- --

shown to the right. If there are 16 -- -- -- -- --

values, then the 16th should print on -- -- --

a fourth line, and so on.

C. The method **findAverage** will receive two parameters, an integer *k* and an array of doubles called *p*. It will find and return the average of the first *k* elements of the *p* array.

D. The method **howFarAway** will receive four parameters: an integer *m,* the average of the array as calculated in the average method, and two arrays of doubles called *r* and *s*. (Here *r* is the original array and *s* is the new array to be constructed. Alternately, you can return the new array *s* instead of passing it as a parameter.) Each element in the new array *s* will equal the difference between the original array’s average and the corresponding element in the original array *r*.

For example, if the original array holds 5.0 3.0 2.0 8.0, then the average is 4.5; the new array will hold 0.5 -1.5 -2.5 3.5. Note that some of these elements are positive and some are negative.

DATA: Read the data from a file which is to be prepared in advance (be sure to print the file). Use a parameter value of 12-15 (but write a program which will work for any size arrays up to 50 or so). Make sure that the numbers are not in order, and make sure that they are spread out. It is not necessary to have decimal places in every value, but a few should. Include both positive and negative values. **PRINT A COPY OF THE DATA FILE**.

Also, all output should be directed to an output file, not the screen.

OPTIONAL Call a method to find the value which is closest and furthest from the average. The method should print these values and how far from the average they are. Remember that you may have positive or negative values.