CMPSCI 111L – Intro to Algorithms & Programming: Java Lab

Programming Project #5 – Arrays and Sorting (30 points)

Due: 11/15/18

Write a Java class named `ScoreStatisticalAnalyzer` containing a `main()` method that reads **double**-type test scores from a data file into an array. The program in your `main()` method should open the file for input, read the *first* number from the file, an **int**, then store this number in an **int**-type variable named **arraySize** (if you were not present for my demonstrations on how to open and read data from text files, refer to section 12.11 on page 480 of the textbook). It should be fairly obvious that this number is an exact count of the remaining **double**-type numbers in the file. The program in your `main()` method should then declare a **double**-type array, named **scores**. Use the **arraySize** variable to specify the size of the array. Read the remaining **double**-type numbers from the file into the **scores** array storing each number in an element of the array.

Once the array is fully "populated", have your main method invoke a named `sort()`, passing the **scores** array to this `sort()` method. This `sort()` method sorts the array into ascending numerical order. You may copy the `selectionSort()` method given on page 272 of the textbook, just be sure to shorten the method name to `sort()`.

After the `sort()` method has done its work, have the program in your `main()` method pass the **scores** array to a method named `averageArray()` which computes and returns the **double**-type average of all the numbers in the array. The `averageArray()` method should have one parameter, a **double**-type array reference variable named **list**. Remember to assign the value returned by `averageArray()` to a **double**-type variable.

After the `averageArray()` method has done its work, have the program in your `main()` method open a text file named **statisticalOutput.txt** for output (if you were not present for my demonstration on how to open and write data to text files, refer to section 12.11 on page 480 of the textbook). Output the **arraySize** first, followed by the sorted **double**-type numbers in the **scores** array. For any scores which are below the average, output an asterisk (*) after the score. Finally, output the average score. Be sure to **close** the file when the program is done.

Use the included **numbers.txt** data file, which contains only 12 scores, to get your program working. Once you are sure your program is working perfectly, change the input filename to scores.txt. This file contains 100 scores, but if you've written the program correctly, it should work in just the same way on this larger file.

Here is what the content of the **statisticalOutput.txt** file should look like after the program is run using **numbers.txt** as input:

```
12
17.05*
26.38*
26.48*
28.65*
41.94*
42.72*
50.78*
58.29
60.48
92.7
92.93
95.22
The average of these scores is: 52.80166666666668
```

If the number of significant digits in the average disturbs you, you may round it to two decimal places by using a special version of the print method called **`printf()`**, to write the average to your output file. The syntax for this "print formatted" output method is:

```
printf("%3.2f ", averageScore)
```

Where %f is a format specifier (a placeholder) for a floating-point number. 3 specifies the width of the number preceding the decimal point, and 2 specifies the number of significant digits that follow the decimal point. If the floating point number has more than two significant digits, the number will be rounded to two significant digits.

Remember to write a Universal comment header at the top of your source code file.

When you have completed the exercise, **ZIP** the entire project folder and upload this **ZIP** file to Canvas. In **NetBeans**, click **File**, select **Export Project**, click **To ZIP...**, select the location where you want the **ZIP** file to be saved, type the name of the **ZIP** file, **Project5.zip** in the **File Name** textbox, click the **Save** button, then click the **Export** button. If the IDE you are using does not have an export-project-to-zip option, you may have to manually **ZIP** the file by navigating to the project folder in a file browser and selecting your OS's "compress folder" option. Under Mac OS, the file browser is called **Finder** which offers a **Compress Folder** option. Under Windows, the file browser is called **File Explorer** and it offers a **Send to=>Compressed(zipped) folder** option.