Wentworth Institute of Technology
COMP1000 – Computer Science I
Fall 2017, Kreimendahl

Programming Assignment 2

In this assignment you are to write programs to solve the following problems. As with all assignments, remember the following submission steps:

- Make sure your code passes at least all the provided JUnit tests
- Create and test Javadoc code documentation
- Save, commit, and push all code changes
- Confirm the latest code is visible via the "Files" section of your repository website
- Confirm that the repository is private, and that the instructor has Developer access

Problem a (`PA2a.java`)

Write a program that reads in five whole numbers from the user. The program will calculate and output the following statistics about the five numbers:

- The sum of all of the positive numbers (greater than zero).
- The sum of all of the non-positive numbers (less than or equal to zero).
- The sum of all five numbers.
- The average of all of the positive numbers (greater than zero).
- The average of all of the non-positive numbers (less than or equal to zero).
- The average of all five numbers.

Note that your program must accept the five numbers in any order and cannot ask the user to enter the positive and non-positive numbers separately. You should start by calculating the sum and average of all five numbers. Be sure you get that working correctly and then move on to doing the calculations for the positive and non-positive numbers. You will need many variables to keep track of all the values, sums, and averages.

Don't forget that to calculate the average of a set of numbers you sum them all up then divide by the number of values you added together. For example, if your program gets 3 positive numbers and 2 non-positive numbers, then you will divide the sum of the positive numbers by 3 to get the average and then divide the sum of the non-positive numbers by 2 to get the average of those numbers. This means you will have to keep track of how many positive numbers you read in and how many non-positive numbers you read in.

Also note that you will need to write a lot of very similar looking pieces of code. Think about how to solve the problem for one or two numbers and then go from there to calculating the statistics for all five numbers.

Finally, for sums, you should output just the number (e.g. 5 or 10) without any decimals (the sum of a set of integers is a sum!). However, for ***all*** averages, you must format the output to have two decimal places (even if they are both 0), and round if necessary.

The following represents a sample run of the program in which the user inputs on a single line "-1 1 -2 2 -3" (without quotes). Note that the JUnit tests, and grading, will be very picky about exact spacing, spelling, capitalization, and number formatting.

```
Enter five whole numbers: -1 1 -2 2 -3
The sum of the 2 positive numbers: 3
The sum of the 3 non-positive numbers: -6
The sum of the 5 numbers: -3
The average of the 2 positive numbers: 1.50
The average of the 3 non-positive numbers: -2.00
The average of the 5 numbers: -0.60
```

Problem b (`PA2b.java`)

Write a program that solves a quadratic equation of the form: $ax^2 + bx + c = 0$

Specifically, the user will input the values of *a*, *b*, and *c*. Your program must then calculate and output the roots of the equation, following these rules.

First, calculate the discriminant: $d = b^2 - 4ac$

- If the discriminant is negative, the roots are imaginary. Print out a message informing the user of this and stop the program. For example:

  ```
  Enter a b c: 1 1 4
  Roots: imaginary
  ```

- If the discriminant is exactly zero, there is one root. Inform the user of this one root: $-b/2a$. You must provide exactly two decimal places in your response, rounding as necessary. For example:

  ```
  Enter a b c: 1 4 4
  Root: -2.00
  ```

- If the discriminant is positive, there are two roots. Inform the user of the value of the roots in increasing order of value (i.e. the smaller root first, followed by the larger). You must provide exactly two decimal places in your response, rounding as necessary. For example:

  ```
  Enter a b c: 1 6 5
  Roots: -5.00, -1.00
  ```

Note that the JUnit tests, and grading, will be very picky about exact spacing, spelling, capitalization, and number formatting.