CS102 – Algorithms and Programming II
Lab Programming Assignment 1
Spring 2022

ATTENTION:

- Feel free to ask questions on Moodle on the Lab Assignment Forum.
- Compress all of the Java program source files (.java) into a single zip file
- The name of the zip file should follow the below convention:

  CS102_SecX_Asgn1_YourSurname_YourName.zip
- Replace the variables "YourSurname" and "YourName" with your actual surname and name and X with your Section id (1, 2 or 3).
- Upload the above zip file to Moodle by February 22nd, 23:59 with at least Part 1 completed. Otherwise, significant points will be reduced. You will get a chance to update and improve your solution (Part 1 and Part 2) by consulting to the TA during the lab. You will resubmit your code (Part 1 and Part 2 together) once you demo your work to the TA.

The work must be done individually. Codesharing and copying code from any source are strictly forbidden. We are using sophisticated tools to check the code similarities. We can even compare your code against online sources. The Honor Code specifies what you can and cannot do. Breaking the rules will result in a disciplinary action.

Part 1

In this lab, you are going to implement an **Equation** class that represents equations of the form a = bx + c

The class should include the following:

1. Assume a, b and c are integers.

2. Include a constructor that takes the coefficients of the linear equation a, b and c and sets the instance variables accordingly. The constructor should ensure that coefficient b is always non-negative. So, if the coefficient of x (b) is negative, you should multiply all of the coefficients by -1. For example:

e.g., -2 = -3x - 1 => 2 = 3x + 1

3. Include a member method **void reduceEquation()** that simplifies the equation. i.e.,

10 = 5x + 5 should be stored as 2 = x + 1.

To simplify the coefficients, you should first find the greatest common divisor of them. For example, your values are a, b and c. Let's assume that the gcd(a , b , c) = s. The simplified version of the given equation will be shown as:

$$\frac{a}{s} = \frac{b}{s}x + \frac{c}{s}$$

4. Implement a private **gcd** method that finds the greatest common divisor of two non-negative integers. Mathematically, the gcd of two numbers is defined as the greatest integer that divides both numbers. Use the Euclid's algorithm to implement gcd (See Box 1).

gcd(20, 15) = 5
gcd(90, 45) = 45
gcd(8, 3) = 1

Box 1. Euclid's Algorithm

Euclid's algorithm is a way of calculating the gcd of two numbers. First, let's describe it using an example. We will find the gcd of 36 and 15.
Divide 36 by 15 (the greater by the smaller), getting 2 with a remainder of 6.
Then we divide 15 by 6 (the previous remainder) and we get 2 and a remainder of 3.
Then we divide 6 by 3 (the previous remainder) and we get 2 with no remainder.

The last non-zero remainder (3) is our gcd. Here it is in general:

a/b gives a remainder of r
b/r gives a remainder of s
r/s gives a remainder of t
...
w/x gives a remainder of y
x/y gives no remainder

In this case, y is the gcd of a and b. If the first step produced no remainder, then b (the lesser of the two numbers) is the gcd.

5. Implement a private method gcd3 that takes three integers and finds their greatest common divisor. This method generalizes the former method gcd to three arguments by calling it. **Hint**: Mathematically gcd(a,b,c) = gcd(gcd(a,b), c).

Part 2

1. Implement two member methods for basic calculations on linear equations.

- **Equation add(Equation eq2)** sums two linear equations (one is the equation for which the method is called (implicit parameter) and the other is eq2) and return the result as a new equation in reduced form.

- **Equation subtract(Equation eq2)** subtracts eq2 from the equation for which the method is called on (implicit parameter) and returns the result as an equation in reduced form.

Summation of equations: (3 = 2x + 1) + (2 = 3x – 1) = (5 = 5x) => (1 = x)
Subtraction of equations: (2 = 3x – 4) – (7 = 4x – 1) = (-5 = -x – 3) => (5 = x + 3)

2. Include a member method called **toString()** method that returns a string representation of the equation.

Suppose that the equation format is **a = bx + c**.

- If b is 0, then you should return "a = c".
- If c is 0, then you should return "a = bx".

3. Inside the **Equation** class *override* the **equals** method of Object class (i.e. **bool equals(Equation eq)**) method that returns true if the implicit parameter equation (i.e. **this)** is equal to **eq**. Please consider the fact that two equations are equal if their reduced versions are equal. For instance, **10 = 5x + 6**, should be equal to **20 = 10x + 12**. However, you have the following constraint. This method should not modify the original equations in the **eq** and **this** variable. That is, you should find ways to compare the reduced versions without actually changing the original equations.

4. Implement a class called **EquationTester** to test your Equation class. You should get the coefficients of the equation from the user on a single line as "2 3 -1.

Example test cases:

2431 = 102 x + 595

208 = -368 x + 1276

-7038 = 2646 x + 558

28 = 3 x + 25

The expected results (reduced equation):

143 = 6 x + 35

-52 = 92 x - 319

-391 = 147 x + 31

28 = 3 x + 25

5. Test the addition and subtraction operations. You may prompt the user to enter a, b and c values for two Equation objects. Example input output are given

Sample Runs: (User inputs are shown in **red**.)

```
Enter the value of a, b and c for first equation: 3 2 1
Enter the value of a, b and c for second equation: 2 3 -1
Sum of the equations: 1 = x

Enter the value of a, b and c for first equation: 2 3 -4
Enter the value of a, b and c for second equation: 7 4 -1
Subtraction of the equations: 5 = x + 3
```

IMPORTANT NOTES:

1. Please comment your code according to the documentation and commenting conventions used in the textbook.