Northeastern Illinois University

CS207 Programming II, Fall 2018 Homework 4

Due: Friday October 19, 2018 at 10:00 pm

Problem 1 BigIntegers (30 points):
Create a class named `LargeIntegers.java`. Include in this class a method as described by the following signature:

```
public void oneOfThree( BigInteger a, BigInteger b, BigInteger c )
```

This method, when called with three `BigInteger` parameters, will output the *larger* of `a` and `b`, *unless* `c` is even, in which case the program outputs the *smaller* of `a` and `b`. However, if `a` and `b` are the same number the program outputs the product of all three *regardless* of whether `c` is even or odd. Use the `compareTo()` method (or others) in class `BigInteger` to make comparisons. Note that you may *not* assume that `a`, `b`, or `c` are small enough to be converted to regular int types.

The table below shows what the program should print with different values of `a`, `b`, and `c`.

a	b	c	Program prints:
13	42	101	Output: 42
13	7	5	Output: 13
10	10	9	Output: 900

Instructions:

- Download the needed files archive and look for `TestLargeIntegers.java`
- Do not modify `TestLargeIntegers.java`
- Run `TestLargeIntegers.java` (make sure your class is in the *same* folder)
- If you created `LargeIntegers.java` correctly, your output should look exactly as the output below

```
Output: 42
Output: 13
Output: 50
Output: 600
Output: 900
Output: 250
```

Problem 2 BigDecimal (30 points):
Create a class named `LargeDecimals.java`. Include in this class a method as described by the following signature:

```
public BigDecimal maxOfMins( BigDecimal[][] table )
```

This method, when called with a 2D array of type `BigInteger`, will return the *maximum* of the *minimum* values in each row. For example, in the array below, the minimum values in each row are `20.0`, `33.0`, `15.0`, and `28.0`. Since the largest of these values is `33.0` `maxOfMins()` will return `33.0` as a `BigDecimal`.

	0	1	2	3	4
0	50.0	20.0	80.0	90.0	100.0
1	77.0	555.0	33.0	44.0	66.0
2	888.0	15.0	302.0	90.0	123.0
3	31.0	29.0	30.0	32.0	28.0

Use the `compareTo()` method (or others) in class `BigDecimal` to make comparisons. Note that you may *not* assume that the values in table are small enough to be converted to regular double types.

Instructions:

- Download the needed files archive and look for `TestLargeDecimals.java`
- Do not modify `TestLargeDecimals.java`
- Run `TestLargeDecimals.java` (make sure your class is in the same folder)
- If you created `TestLargeDecimals.java` correctly, your output should match the output below

```
The max of the mins in numbers1 is: 33.0
The max of the mins in numbers2 is: -14.0
```

Problem 3 Strings (40 points):
Open the `String` class documentation in the Java API, and take a look at the different versions of methods `indexOf()` and `substring()`. These methods are overloaded.

Identify what parameters each version of those methods takes and what is returned, then do the following:

Create a class named `MiniString.java`, which has the following:

- A *private* instance variable of type String named `str`
- A constructor that accepts a String as a parameter and sets the instance variable to it
- Two methods, both named `makeSubstring()`, each of which will implement one of the overloaded versions of `substring()` in the Java API
- Four methods, all named `findIndexOf()`, each of which will implement one of the overloaded versions of `indexOf()` in the Java API
- The only 3 methods from class String you can use to solve this problem are: `charAt()`, `length()` and `equals()`
- IF YOU use any other methods from the String class, you will not get credit for this problem.

In this problem, you are actually re-inventing the wheel. You are implementing methods that already exist in the Java API. However, it is really important to know how those methods work, and how to build them from scratch. Remember, Strings in Java are *very* similar to 1-D arrays from Programming 1.

Instructions:

- Locate the `TestStringPractice.java` file from the Needed Files and open it
- To test each method, uncomment the section that uses that method in `main()`, and make sure that the output matches the sample output shown on the next page

```
Hello world, welcome to the Java world.
world, welcome to the Java world.
world, w
welcome to the Java world.
Hello world,

6
18
-1
28
29

13
-1
-1
-1
31

6
13
0
28
30

33
-1
-1
-1
30
```

General Instructions:

No hard copies will be collected. Do not send your files through e-mail! You should submit your work on D2L by the due date. See syllabus for late homework policy.

What to turn in:

There should be three Java files: `LargeIntegers.java`, `LargeDecimals.java`, and `StringPractice.java`. Place these three files into a zip file and name it *`YourName`*`HW4.zip`, and submit the zip file to the **Homework 4** folder in D2L. Do **not** turn in `.class` files.

Helpful Links:

How to zip files in Mac OS

How to zip files in Windows