Programming Assignment #1
ADT for Very Long Integers

Problem Statement

The `java.math.BigInteger` class of Java can represent arbitrary-precision (*i.e.*, unbounded) integers and provides methods to perform arithmetic operations on them. For this programming assignment, we design a similar abstract data type (ADT) called `LongInt`, which defines a subset of the methods provided by the `java.math.BigInteger` class.

Requirements & Implementation

- You are to write a Java application that implements the `LongInt` ADT. A skeleton of the ADT is provided in the eTL site. The skeleton file is named `LongIntSkel.java`. You can copy it to your working directory and rename it to `LongInt.java`. You should then provide implementations of all the methods defined in the class skeleton. You can add more methods as you wish, but are not allowed to remove any method defined in the class skeleton.
- The goal of this assignment is to learn how to build an ADT using primitive data types. You are not allowed to use the `java.math.BigInteger` or `java.math.BigDecimal` class to implement the `LongInt` ADT.
- No `main()` method is necessary in your class implementation. We will test and evaluate your class implementation with our own test program (`MainInt.java`) containing the `main()` method, which is available in the eTL site. This test program uses `TextInputStream` class for input and output, which is also provided. So, there is no need to worry about I/O and no need to turn in any additional class containing a `main()` method.
- Once you have compiled your classes and `MainInt.java` successfully, you can type the following at the Unix shell prompt to run and test your code.

  ```
  % java MainInt test1
  ```

  Sample input files (*e.g.*, `test1`) and their output files (*e.g.*, `test1.out`) are provided in the eTL site.
- Each input file has one or more lines each of which contains two decimal integers connected by an arithmetic operator. Assume input integers are *well formed* in the sense that all characters are numeric between 0 and 9 except a leading minus sign and there are no leading zeros or commas. The output integers must be well formed too.

Grading

This assignment is worth 5 percent of your final grade. Efficiency is not considered for grading this assignment. However, if your program does not terminate within a reasonable amount of time (*e.g.*, 100 times the average runtime of the class) and needs to be aborted, it will be considered incorrect. General grading guidelines are:

10 points	Program compiles without errors and is on the right track,
0-50 points	Works on *simple* test cases,
0-40 points	Works on *complex* test cases.

The late submission and regrading policies are described in the course syllabus.

Submission

Turning in your programming assignment must be done at the eTL site. Do not submit any `.class` or test files. Submit only `.java` files in a single archive (*e.g.*, zip or tar). Refer to the course syllabus for general submission instructions. More specific instructions will be given by the TA.

Due date

The programming assignment is handed out on Wednesday September 07, 2016, and due by 11pm on Saturday September 24, 2016. (Due to the Thanksgiving holidays, the due date of this assignment is pushed back to Saturday in order to give your more time.)