CS/COE 1501 Assignment 5

Released: Friday, April 10

Due: Friday, April 17, 11:59 PM

Goal

To get hands on experience with algorithms to perform mathematical operations on large integers.

High-level description

You will be writing a replacement for Java's `BigInteger` to perform multiplications and to run the extended Euclidean algorithm on integer values that would overflow `long`.

Specifications

1. You are provided with the start of a class to process arbitrarily-sized integers called `HeftyInteger`. `HeftyInteger` objects are represented internally as two's-complement *raw integers* using byte arrays (i.e., instances of `byte[]`).

    1. Currently, `HeftyInteger` has the following operations implemented:

        - A constructor that creates a new `HeftyInteger` object based on a provided `byte[]`.
        - A method to compute the sum of two `HeftyInteger` objects.
        - A method to determine the negation of a `HeftyInteger` object.
        - A method to compute the difference of two `HeftyInteger` objects.
        - Several other helper methods.

    2. Due to the use of a two's complement representation of the integers, positive `HeftyInteger` objects should always have at least one leading 0 bit (indicating that the integer is positive) in their `byte[]` representation. This property may cause the array to be bigger than expected (e.g., a 1024-bit positive integer will be represented using a length 129 byte array).
    3. HeftyIntegers are represented using a *big-endian* byte-order, so the most significant byte is at index 0 of the `byte[]`.

4. You will further need to implement the following functions:
    - `HeftyInteger multiply(HeftyInteger other)`
    - `HeftyInteger[] XGCD(HeftyInteger other)`
    - Any additional helper functions that you deem necessary.

5. You may *not* use any calls the Java API class `java.math.BigInteger` or any other JCL class within `HeftyInteger`.

2. Once `HeftyInteger` is complete, make sure your implementation of `HeftyInteger` can be used to run the driver programs contained in `MultiplicationTest.java` and `XgcdTest.java`. To get full credit, your implementation should be efficient enough to complete multiplication or XGCD given 200-digit inputs within 3 minutes.

Submission Guidelines

- **DO NOT** upload any IDE package files.
- You must be able to compile the driver programs by running `javac MultiplicationTest.java` and `javac XgcdTest.java`, respectively.
- You must be able to run the driver program by running `java MultiplicationTest` and `java XgcdTest`, respectively.
- You must fill out `info_sheet.txt`.
- The project is due at the precise date and time stated above. Upload your progress to Box frequently, even far in advance of this deadline. **No late assignments will be accepted.** At the deadline, your Box folder will automatically be changed to read-only, and no more changes will be accepted. Whatever is present in your Box folder at that time will be considered your submission for this assignment—no other submissions will be considered.

Grading Rubric

`HeftyInteger`

Feature	Points
`multiply`	40
`XGCD`	55
Assignment info sheet/submission	5