CS102 - Algorithms and Programming II
Lab Programming Assignment 5
Fall 2016

ATTENTION:

- Feel free to ask questions on Moodle on the Lab Assignment Forum.
- Compress all of the Java program source files (.java) files into a single zip file.
- The name of the zip file should follow the below convention:
  CS102_SecX_Asgn5_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 the deadline before the lab (if not significant points will be taken off). You will get a chance to update and improve your solution by consulting to the TA during the lab. You will resubmit your code once you demo your work to the TA.

GRADING WARNING:

- Please read the grading criteria provided on Moodle.

Q1. Write a class **`BigInteger`** that keeps a very large positive integer in an integer array called *digits* whereas each digit will be kept in one element of the array. The **constructor** takes the number as a String and adds each digit character by character into the integer array. Make sure you avoid leading zeros (00123 = 123) and the orientation of the **`BigInteger`** is correct. Implement the following methods:

`int numberOfDigits():` returns the number of digits in the BigInteger.

`int MID():` returns the most important digit. The most important digit is the left most digit in a number for example MID of 123456 is 1.

`int LID():` returns the least important digit. The least important digit is the right most digit in a number for example LID of 123456 is 6.

`int getDigit(int index) :` returns the digit at the given index. Make sure a call to **`getDigit(0)`**returns the LID and a call to **`getDigit(numberOfDigits()-1)`**returns the MID.

BigInteger class implements the Comparable interface.

Overrides the Object class equal method.

Write a **`BigIntegerTester`** tester class to check each of these methods work correctly.

Q2. Write **`BigIntegerList`** class that keeps a list of `BigIntegers`. The only attribute of `BigIntegerList` is an ArrayList of BigIntegers called *numbers*. The constructor takes an ArrayList of Strings that represents BigInteger and one by one adds them to the *numbers* list as BigIntegers. The class should have the following methods:

`int getSize()`
`BigInteger getBigIntegerAt(int index)`

```
void setBigIntegerAt(int index, BigIntegerbigInt)
void addBigInteger(String number)
void removeBigInteger(int index)
void removeBigInteger(BigIntegerbigInt)
```

Additionally, implement a method **`min(int start, int end)`** that returns the minimum BigInteger number in the list between the start and end index. You MUST implement this method in a recursive fashion. The simple solution is to compare the first element to the minimum of the rest. Another one is to check the minimum of the first half and compare to the minimum of the second half. What would be the difference?

Q3. Write a `BigIntegerTester` class that reads a file that contains BigIntegers as Strings in each line.Include a public static readBigIntegerFile method to read the user input file. Initialize a BigIntegerList object using these strings. First print the minimum of the entire list. Then print the minimum of the first half, and then the minimum of the second half. Sample input files are provided along with the assignment but think of other interesting cases and test your program also with files that you create. In this class you should also implement the following static method

```
static BigIntegerList getBigIntegersFromFile( String fileName)
```

Sample Outputs

```
Please enter the filename: BigNumbers1.txt
Output:

start index = 0
middle index = 19
end index = 38

Minimum of all the numbers:
2348786224978743588798209309409435878782364344794509123867635982467600
2348786224978743588798209309409435878782364344767945091238676359824676
88798209309409435878782364344767
Minimum of the first half:
2348786224978743588798209309409435878782364344767945091238676359824676
0023487862249787435887982093094094358782364344767945091238676359824676
8879820930940943587878236434476712
Minimum of the second half:
2348786224978743588798209309409435878782364344794509123867635982467600
2348786224978743588798209309409435878782364344767945091238676359824676
88798209309409435878782364344767
```

```
Please enter the filename: BigNumbers2.txt
Output:

start = 0
```

```
middle = 499
end = 999
Minimum of all the numbers:
700
Minimum of the first half:
700
Minimum of the second half:
806
```

```
Please enter the filename: BigNumbers3.txt
Output:

start = 0
middle = 2
end = 4
Minimum of all the numbers:
21
Minimum of the first half:
21
Minimum of the second half:
41
```