CS102 - Algorithms and Programming II
Lab Programming Assignment 2
Spring 2022

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_Asgn2_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 March 1st, 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

Write a class **IntegerArray** 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 **IntegerArray** is correct. Implement the following methods:

- **int numberOfDigits():** returns the number of digits in the **IntegerArray**.
- **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.
- **IntegerArray add(IntegerArray other):** returns the sum of **this IntegerArray** and **other IntegerArray**. Here, you should develop algorithms to sum up two arrays each representing a number.
- **IntegerArray subtract(IntegerArray other):** returns the subtraction of **other IntegerArray** from **this IntegerArray**. You should use above **add** method while implementing this one.
- **IntegerArray** class implements the Comparable interface. Overrides the Object class equal method.
- Write a **IntegerArrayTester** tester class to check if each of these methods work correctly.

Part 2

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

- **int getSize():** returns the number of **IntegerArray**s in the list
- **IntegerArray getIntegerArrayAt(int index)**: returns the IntegerArray at the given index.
- **void setIntegerArrayAt(int index, IntegerArray intArr):** sets the element at given index.
- **void addIntegerArray(String number):** adds a new **IntegerArray** to the ArrayList.
- **void removeIntegerArray(int index):** remove the IntegerArray at given index.
- **void removeIntegerArray(IntegerArray intArr):** removes the given IntegerArray object from the list.
- Additionally, implement a method **IntegerArray min(int start, int end)** that returns the minimum **IntegerArray** number in the list between the start and end index.
- Expand **IntegerArrayTester** class so that it reads a file that contains **IntegerArray**s as Strings in each line. Include a **public static IntegerArrayList readIntegerArraysFromFile(String fileName)** method to read the user input file. Initialize a **IntegerArrayList** 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. To read the files in Java, you can use the code snippet in this link :

  https://www.w3schools.com/java/java_files_read.asp

Sample Outputs

```
Please enter the filename: Test1.txt
Output:

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

Minimum of all the numbers:
234878622497874358879820930940943587878236434479450912386763598246760 0
234878622497874358879820930940943587878236434476794509123867635982467676
887982093094094358787823643447679
Minimum of the first half:
234878622497874358879820930940943587878236434476794509123867635982467676
002348786224978743588798209309409435878236434476794509123867635982467676
887982093094094358787823643447671 2
Minimum of the second half:
234878622497874358879820930940943587878236434479450912386763598246760 0
234878622497874358879820930940943587878236434476794509123867635982467676
887982093094094358787823643447679
```

```
Please enter the filename: Test2.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: Test3.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
```