Computer Laboratory 4
CSci 1913: Introduction to Algorithms,
Data Structures, and Program Development
February 14/15, 2017

0. Introduction.

This laboratory assignment is intended as a short introduction to Java for students who are already familiar with C or C++. It asks you to rewrite the Python class `Zillion` from Lab 2 in Java. Like the Python class, the Java class `Zillion` implements a decimal counter with an arbitrarily large (but bounded) number of digits. However, since you will be using a Java array instead of a Python list, your counter will have a fixed maximum number of digits.

1. Example.

All the code in this example goes inside the `main` method of your driver class. It first creates an instance of `Zillion` in the variable `counter`, like this.

```
Zillion counter = new Zillion(12);
```

The instance of `Zillion` can hold up to twelve decimal digits. Its initial value is 000000000000. Note that twelve digits can represent a much larger number than will fit in a 32-bit Java `int`, which is only 2 147 483 647, around two billion. The `main` method then adds 1 to the counter by calling the method `increment`, like this.

```
counter.increment();
```

Now the counter contains 000000000001. The `main` method now prints the counter like this. Java automatically calls `counter`'s `toString` method when you try to print it.

```
System.out.println(counter);
```

You should see `000000000001` printed. If you call `increment` again, and then print the counter, then you should see `000000000002`. If you call `increment` and print the counter one more time, then you should see `000000000003`, etc.

2. Theory.

Your class `Zillion` must represent the digits of a number using a private array of integers (`int`'s). Each digit d in the array must be an integer $0 \le d \le 9$. For example, suppose that your array is called `digits`, has a length of 3, and represents the number 57. Then `digits[0]` is 0, `digits[1]` is 5, and `digits[2]` is 7.

The method `increment` must work like this. Starting at the right end of the array, and moving toward the left end, it must change 9's into 0's, until it finds a digit that is not 9, or until there are no more digits left to be visited. If it stops because it has visited a digit that is not 9, then it must add 1 to that digit. If it stops because there are no more digits left, then the counter has overflowed and it must not do anything else. For example, if your counter has three digits, and represents the integer 999, then incrementing it must produce 000. It must not indicate an error in any way.

3. Implementation.

The class `Zillion` must contain the following methods. To simplify grading, your methods must use the same names as these. However, they need not use the same parameter names.

```
public Zillion(int size)
```

> Constructor. Create a private array of `size` integers (`int`'s). You may assume that `size` is greater than or equal to 0. You need not initialize the array, because Java initializes all the

elements to 0 for you.

`public void increment()`

Increment the counter, using the algorithm described in part **2.** One way to do that is by using a `while` loop; there may be other ways. If the counter contains all 9's, then you must not generate an illegal array index.

`public String toString()`

Convert the digits in the counter to a `String`, and return that `String`. One way to do that is by using a `for` loop that concatenates all the digits of the array together using the operator `+`. Note that Java automatically converts an integer to a string when you concatenate it.

4. Tests.

The file **`Driver.java`** contains Java source code for a driver class, unimaginatively called `Driver`. This class contains a `main` method, which in turn contains code that tests your class `Zillion`. Note that the class `Zillion` must *not* contain the `main` method! To grade your work, the TA's will run the `main` method. If a test succeeds, then you will receive all the points for that test. If a test fails, then you will receive no points for that test.

Starting with this lab assignment, the TA's may also perform *secret tests* that you have not been told about. They will use the same secret tests for all students. If a secret test fails, then you will lose points for it. You will be told what the secret tests were, and how many points they were worth, after this assignment has been graded.

4. Deliverables.

You must turn in Java source code for your class `Zillion`. Do not submit the results of your tests. Your lab TA will tell you how and where to turn in your work. If your lab is on Tuesday, February 14, then your work must be submitted by Tuesday, February 21 at 11:55 PM. If your lab is on Wednesday, February 15, then your work must be submitted by Wednesday, February 22, at 11:55 PM. *To avoid late penalties, do not confuse these two dates!*