Lab 9 -- Look-and-Say

The goal of this lab is to design and implement an Iterator design pattern.

Look-and-say sequences

A look-and-say sequence is a sequence of numbers. Given the current number, the next number is obtained by reading the current number out loud, writing what we say. For example, if the current number is "1", then we read that as "one 1" and thus the next number in the sequence will be "11" . Similarly, if the current number is "112321", then we read that as "two 1s, one 2, one 3, one 2, one 1", producing the next number as "2112131211", and so on. A look-and-say sequence is a simple example of run-length encoding.

The look-and-say sequence can also be computed in reverse. For example "2112131211" can be reversed by taking digits two at a time, and writing them out (write two 1s, then one 2, and so on). This produces "112321". Work out a couple of examples of the sequence before proceeding.

What to do

**Package:** lookandsay

[Download this RIterator interface](https://northeastern.instructure.com/courses/63372/files/8485435?wrap=1)[download](https://northeastern.instructure.com/courses/63372/files/8485435/download?download_frd=1)that extends the standard Iterator interface and adds two methods: T prev() and boolean hasPrevious(). Their meanings are self-explanatory. Then define a class LookAndSayIterator that implements RIterator. This iterator will return the numbers as BigInteger objects.

The LookAndSayIterator class should have the following characteristics:

1. It must offer a constructor that takes two arguments: a starting seed and an end value. The seed is the number at which the sequence must begin. **The iterator should not produce a number greater than the end value.** If the seed is not positive, or is not less than the end, or has any zeroes in it, the constructor should throw an IllegalArgumentException.
2. It must offer a constructor that takes a starting seed as its only argument. The seed is the number at which the sequence must begin. The end value will be a number with 100 9s (the largest 100 digit number). If the seed is not positive, or is not less than the end, or has any zeroes in it, the constructor should throw an IllegalArgumentException.
3. It must offer a constructor that takes no arguments. This should start the sequence with a seed of 1 (e.g. calling next and an end value as a number with 100 9s (the largest 100 digit number).
4. next() should return the current number in the sequence and advance to the next number. If there is no next value in the sequence, this method throws a NoSuchElementException.
5. prev() should return the previous number in the sequence, that is, the number coming before the last number returned by next(). If there is no previous value in the sequence, this method throws a NoSuchElementException.
6. hasNext() should return true if the next number to be returned is still lesser than end (specified in the constructors), false otherwise.
7. hasPrevious() should return true if it is possible to go back one step, false otherwise. **How would you determine this?**

Note that according to the above specification, the sequence may begin in the middle of a sequence (depending on what the seed is) but be able to go back behind the seed, until it cannot. For example, if the seed is "11" , it is possible to go backward beyond "11" to "1" .