Introduction to Computer Programming (CS 1323)
Project 4

Instructor: Andrew H. Fagg

October 12, 2014

Introduction

A *Lucas Sequence* is a sequence of numbers $L_{pq}(0), L_{pq}(1), L_{pq}(2), ...$ that can be constructed using the following rules:

$$\begin{aligned} L_{pq}(0) &= 0, \\ L_{pq}(1) &= 1, \text{and} \\ L_{pq}(i) &= p \times L_{pq}(i-1) - q \times L_{pq}(i-2) \text{ where } i \geq 2 \ . \end{aligned}$$

Depending on the choice of parameters p and q, one can achieve sequences of numbers with different properties. For example, $L_{1,-1}$ is the *Fibonacci* sequence.

Objectives

By the end of this project, you will be able to:

1. Write loops to generate numbers from a simple recurrence relation,
2. create and manipulate **ArrayLists**,
3. search through an ArrayList to find a value that satisfies particular criteria.

Project Requirements

1. Write a **getInt()** method that takes as input a String prompt and a Scanner, and returns an integer. This method prompts a user for an integer. If the entered value is an integer, then it is returned. If it is not an integer, then this method must re-prompt until an integer is entered.

2. Write a **getLucasSequence()** method that takes as input integer parameters p, q and n. Where p and q are Lucas parameters as defined above and n determines how many Lucas numbers are generated. This method returns an ArrayList of integers:

   $L_{pq}(0), L_{pq}(1), L_{pq}(2), ..., L_{pq}(n)$

3. Write a **checkSequence()** method that takes as input an ArrayList of integers, identifies the first i that is greater than or equal to 2 for which $L_{pq}(i) \geq i^4$ and prints out:

   $i : L_{pq}(i)\ i^4$

   If there is no such i, then this method must print out "None found"

4. Write a **main()** method that uses the methods you have written to receive p, q and n from the user, prints out the resulting Lucas Sequence and then calls checkSequence().

Examples

Here are some example inputs and corresponding outputs from a properly executing program (note that user interaction output is not shown). You should carefully consider the cases with which you test your program. In particular, think about the key "boundary cases" (the cases that cause your code to do one thing or another, based on very small differences in input).

Example 1

Input:

```
3
1
20
```

Output:

```
[0, 1, 3, 8, 21, 55, 144, 377, 987, 2584, 6765, 17711, 46368, 121393,
317811, 832040, 2178309, 5702887, 14930352, 39088169, 102334155]
11: 17711 14641
```

Note: within the brackets, there should not be any newlines.

Example 2

Input:

```
2 foo 2
20
```

Output:

```
[0, 1, 2, 2, 0, -4, -8, -8, 0, 16, 32, 32, 0, -64, -128, -128, 0, 256,
512, 512, 0]
None found
```

Example 3

Input:

```
bar 2
baz -1 foobar 30
```

Output:

```
[0, 1, 2, 5, 12, 29, 70, 169, 408, 985, 2378, 5741, 13860, 33461, 80782,
195025, 470832, 1136689, 2744210, 6625109, 15994428, 38613965, 93222358,
225058681, 543339720, 1311738121, -1128151334, -944564547, 1277686868,
1610809189, 204337950]
13: 33461 28561
```

Project Documentation

Follow the same documentation procedures as we used in project 3. In particular, you must include:

- Java source file documentation (top of file)
- Method-level documentation
- Inline documentation

For full credit, you must execute Javadoc on your source file and include the results (in the *doc* directory) in your zip file.

Hints

Implement and test your program one method at a time (we call this *incremental development*). Convince yourself that each method is working properly before you move on to the next one.

Project Submission

This project must be submitted no later than 2:00pm on Thursday, October 16th to the project 4 D2L dropbox. The file must be called *Project4.zip* and be generated in the same way as for Project 3.

Code Review

For the office hour sessions surrounding the project deadline, we will put together a "Doodle Poll" through which you can sign up for a 5-minute code review appointment. During this review, we will execute test examples and review the code with you. If you fail to show up for your reserved time, then you will forfeit your appointment and you must attend at a later time that is not already reserved by someone else.

If either the instructor or TA are free during office hours (or another mutually-agreed upon time), then we are happy to do code reviews, as well as provide general help.

We will only perform reviews on code that has already been submitted to the dropbox on D2L. Please prepare ahead of time for your code review by performing this submission step.

Code reviews must be completed no later than Monday, October 27th. If you fail to do a code review, then you will receive a score of zero for the project.

Anyone receiving a 95 or greater on Project 3 may opt to do an "offline" code review (which does not require your presence). Send the instructor email once you have submitted your project in order to schedule an offline review.

Rubric

The project will be graded out of 100 points. The distribution is as follows:

Implementation: 35 points

Program formatting: 15 points

(15) The program is properly formatted (including indentation, curly brace and semicolon locations).

(8) There is one problem with program formatting.

(0) The program is not properly formatted.

Data types and method calls: 10 points

(10) The program is using proper data types and method calls.

(5) There is one error in data type or method call selection.

(0) There are multiple errors in data type and method call selection.

Required Methods: 10 points

(10) All of the required methods are implemented correctly.

(0) The required methods are not implemented correctly.

Proper Execution: 30 points

Output: 15 points

(15) The program passes all tests.

(12) The program fails one test.

(9) The program fails two tests.

(6) The program fails three tests.

(3) The program fails four tests.

(0) The program fails five or more tests.

Execution: 15 points

(15) The program executes with no errors (thrown exceptions).

(8) The program executes, but there is one minor error.

(0) The program does not execute.

Documentation and Submission: 35 points

Project Documentation: 5 points

(5) The java file contains all of the required documentation elements at the top of the file.

(3) The java file is missing one of the required documentation elements.

(2) The java file is missing two of the required documentation elements.

(0) The java file is missing more than two of the required documentation elements.

Method-Level Documentation: 10 points

(10) Every method contains all of the required documentation elements ahead of the method prototype, and the Javadocs have been generated and included in the submission.

(7) The method documentation is missing one of the required documentation elements.

(3) The method documentation is missing two of the required documentation elements, or the Javadocs have not been submitted.

(0) The method documentation is missing more than two of the required documentation elements.

Inline: 10 points

(10) Every method contains appropriate inline documentation.

(7) There is one missing or incorrect line of inline documentation.

(3) There are two missing or incorrect lines of inline documentation.

(0) There are more than two missing or incorrect lines of inline documentation.

Submission: 10 points

(10) The correct zip file name is used.

(0) An incorrect zip file name is used.

Bonus: 6 points For each unique and meaningful error in this project description that is reported to the instructor, a student will receive an extra 2 points.

References

- Generalizations of Fibonacci numbers: http://en.wikipedia.org/wiki/Generalizations_of_Fibonacci_numbers

- Lucas sequences: http://en.wikipedia.org/wiki/Lucas_sequence