# CS1083 Assignment #7 – Winter 2021

**Due: Friday, March 12th before 4:00pm (Atlantic), submitted in the Assignment 7 dropbox in Desire2Learn. (Read the submission instructions at the end of this document carefully).**

The purpose of this assignment is to help develop your understanding of recursion.

**This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.**

---

**Note: you will need to download the zip file posted in D2L and modify the files as directed in the assignment.**

## Computation of a Recursive Sequence: three solutions

A recursive mathematical sequence is one in which each term is defined using the previous term. In the sequence 1, 5, 6, 11, 17, ... the first two terms are the seed terms, and each subsequent term is calculated using the sum of the previous 2 terms. More specifically, if seq(n) is the $n^{th}$ term of the sequence, then the sequence can be defined mathematically as follows:

```
seq(0) = 1
seq(1) = 5
seq(n) = seq(n-1) + seq(n-2), n>1
```

a) Because the sequence is defined recursively, it is natural to write a recursive method to determine term number *n* in the sequence. The file `Seq.java` contains the skeleton for a class containing a method to compute the numbers in the sequence. Save this file to your directory. Following the specification above, fill in the code for method `seqR` so that it **recursively** computes and returns term *n* in the sequence.

   The program `TestSeq.java` contains a simple driver that asks the user for an integer and uses the `seqR` method to compute that element in the sequence. Save this file to your directory and use it to test your `seqR` method. Be sure to test for boundary cases such as the first two terms (0 and 1). Test other values of n such

as 4 (the result should be 17), 10 (the result should be 309) and 45 (the result should be 2080957287). What do you notice if you try to find the 60th term in the sequence?

The problem is that the seqR method is making lots and lots of recursive calls. To see this, add a print statement at the beginning of your seqR method that indicates what call is being computed, e.g., "In seqR(3)" if the parameter is 3. Now run TestSeq again and enter 5—you should get a number of messages from your print statement. Examine these messages and figure out the sequence of calls that generated them. This is easiest if you first draw the call tree on paper. Since seqR(5) is seqR(4) + seqR(3), you should not be surprised to find calls to seqR(4) and seqR(3) in the printout. But why are there two calls to seqR(3)? Because both seqR(4) and seqR(5) need seqR(3), so they both compute it—very inefficient.

**Note:** remember to remove the print statement you added to the seqR method, this was just for demonstration purposes.

b) To improve the time that it takes to recursively compute term *n* in the sequence is to use Memoization. Memoization is the process of storing the values in a cache (memory) as they are calculated and returning the cached result when the same input occurs. For our cache we will use an ArrayList. Proceed as follows:

i. Add a method seqM to your Seq class. Like seqR, seqM should be static and should take an integer parameter and return an integer. Add an ArrayList object as a **class variable**.

ii. Inside seqM, check if the ArrayList has been initialized (remember, non-initialized objects will contain a null value by default), if not then you will need to initialize it to the correct capacity. Since the first two values of the sequence are given in the definition, we can add the first two values to the ArrayList depending on the capacity.

iii. Check if the value requested has already been calculated and stored in the cache (ArrayList). If so then it does not have to be recalculated, if not then the value will need to be calculated and stored in the appropriate index of the cache (ArrayList). You may want to review the methods of the ArrayList class that are listed in the API which can be found online.

iv. Modify your TestSeq class so that it also calls seqM and prints the result. You should get the same answers.

(Continued on next page)

c) Another way to compute term *n* in the sequence is to use an iterative approach and store the values in an array as they are computed. Proceed as follows:
   i. Add a method `seqI` to your `Seq` class that is static and takes an integer as its parameter and returns an integer.
   ii. Inside `seqI`, create an array of integers that you will use to store the values of the sequence. Using a loop, compute each element of the array as the sum of the two previous elements (except the first two elements which will be assigned their values) up to the value passed in. When the array is full, its last element is the element requested. Return this value.
   iii. Modify your `TestSeq` class so that it calls `seqR` and prints that result, then calls `seqM` and prints the result, and lastly calls `seqI` and prints that result. You should get all the same answers.

   **Note:** you can always assume that the user will enter a valid input value (ie: non-negative integer) when testing your methods. You do not have to test for or handle cases where the user inputs an invalid value.

d) To determine the time that each method takes to execute we can use the nanoTime() method from the System class and calculate the difference in the time before and after the method call. Produce a nicely formatted printout of values of the average execution times by putting all three method calls and the printout of their execution times inside of a loop. You can use NumberFormat class to keep values lined up in the tab separated columns.

   For testing, use the sample code on the next page in your main method and add the remaining method calls and time calculations to the body of the for loop. Include the output in your report.

(Continued on next page)

```
NumberFormat form = NumberFormat.getInstance();
form.setMaximumFractionDigits(7);
form.setMinimumFractionDigits(7);

System.out.println("Execution Times in Milliseconds (ms)");
System.out.println("Seq(n) \tRecursive \tMemoization \tItertive");

long start, end;
int seqA;
double time;

for(int i = 20; i <= 40; i+=10){
   start = System.nanoTime();
   seqA = Seq.seqR(i);
   end = System.nanoTime();
   time = (double)(end-start)/1000000;

   System.out.print(i + "\t" + form.format(time));

   //add remaining method calls and time calculations

 }
```

**Note:** Code must be appropriately commented (including Javadoc comments).

**Submission instructions are on the next page…**

**For this assignment, only an electronic submission is required.**

**Your electronic submission (submitted via Desire2Learn) will consist of two files:**

i. a written report. This should begin with a title page that includes: the course (CS 1083), your section (FR01B, FR02B, FR03B), the assignment number, your full name, and your UNB student number. That should be followed by two sections, with each part clearly identified with a section heading. Include:

- the source code for `Seq.java` and `TestSeq.java`
- the sample output

This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). Copy & paste your java source code & required output into the report document. Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) **The SINGLE pdf file containing your report will be submitted to the appropriate assignment drop box on Desire2Learn**. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

Note: name this report as follows: **YourName_CS1083_As7_Report.pdf**

ii. an archive file (**.zip**) that contains all your Java source code for this assignment. Make sure that your archive **includes all .java files** (in case the marker wishes to compile & run your code to test it). You should not include the report document or the .class files in your archive. This archive should be submitted as a **single file** to the appropriate drop box on Desire2Learn.

Note: name this archive file as follows: **YourName_CS1083_As7_Archive.zip**