CS 0445 Spring 2022 Assignment 3

Online: Tuesday, March 1, 2022
Due: All files submitted in a single .zip file via the submission site by **11:59PM on Tuesday, March 22, 2022.**
Late Due Date: 11:59PM on Thursday, March 24, 2022

Purpose: Now that we have looked at some interesting recursive programs, we can try some recursive programming for ourselves. In this assignment you will re-implement the LinkedListPlus<T> and ReallyLongInt classes, with the identical methods and data as in Assignment 2. However, now you must use recursive methods for all of your operations where iteration was used previously. This will give you good practice in programming recursively and will help you to better understand designing and implementing recursive methods.

Goal and Details: You will implement classes LinkedListPlus2<T> and ReallyLongInt2, which will have the same headers and the same functionality as LinkedListPlus<T> and ReallyLongInt specified in Assignment 2 (the 2s are only added to eliminate confusion for file names). Thus, your headers will be:

```
public class LinkedListPlus2<T> extends A2LList<T>
```

and

```
public class ReallyLongInt2 extends LinkedListPlus2<Integer>
                            implements Comparable<ReallyLongInt2>
```

However, in the LinkedListPlus2<T> and ReallyLongInt2 classes you **cannot have any loops of any kind in your code**. Any method that previously utilized loops **must now be implemented using recursion**. **No credit will be given for methods that have any loops in them, or that call methods containing loops (see exception in the next paragraph). The same rules about unnecessary copying and traversing apply for these classes – do not do more work than you need to do. The same rules against casting also apply for ReallyLongInt2 – you must act on the digits of the number in the ReallyLongInt2 and cannot cast to some other type to do the math.**

Note: The A2LList<T> class is unchanged from Assignment 2 and thus still has iteration within it. You may still use / call methods from this class in your recursive implementations. However, **you may not have any iteration within your own code**. **Note, however, that you may not use the getEntry() or getNodeAt() methods from A2LList<T> in your classes.**

Some of the methods will be fairly straightforward, especially once you are familiar with a recursive approach. However, the mathematical operations may be a bit tricky due to several cases that must be handled. The multiply() method, in particular should be carefully considered. Think about what your iterative version is doing and how that can be converted into a recursive form. As a hint for this, you are allowed to add some private methods to help with this implementation as long as they are also recursive.

Test your classes with the files LLPTest2.java and RLITest2.java. These main programs are identical to those from Assignment 2 except that LinkedListPlus2<T> is substituted for LinkedListPlus<T> and ReallyLongInt2 is substituted for ReallyLongInt. The output should be identical to that from Assignment 2: LLPTest.txt and RLITest.txt

Hints and Help: Recursive algorithms take some getting used to, and if you are not experienced in using them it may be difficult to even know how to begin. For help getting started with these implementations, **see Recitation Exercise 5 and its solution (the solution is posted on the course Canvas site)**. This has 3 recursive methods on a doubly-linked list and can give you some ideas about how to proceed with your

recursive implementations. Look over this solution and especially the comments in the code. Look at the recursive cases and the base cases and how the execution is proceeding. Think about how these methods are working and trace the execution (you can even add some extra trace code if you wish). You can then apply the ideas from these methods into your own methods in the LinkedListPlus<T> and ReallyLongInt classes.

As mentioned above, your methods are allowed to call other (private) methods within your class, as long as those methods are also recursive. This approach may be particularly helpful in the multiply() method.

Make sure you **submit ALL** of the following in your **single .zip file** to get full credit:

1) All of the files that you wrote, well-formatted and reasonably commented:
   a) LinkedListPlus2.java (recursive version)
   b) ReallyLongInt2.java (recursive version)
   c) Any other Java files that you may have also written
2) All files provided for you, unchanged from how you downloaded them:
   a) A2LList.java
   b) LLPTest2.java
   c) RLITest2.java
3) Your completed Assignment Information Sheet. Be sure to help the TA with grading by clearly indicating here what you did and did not get working for the project.

As with Assignments 1 and 2, the idea from your submission is that your TA can compile and run your programs WITHOUT ANY additional files, so be sure to test them thoroughly before submitting them. If you used an IDE to develop your solution, make sure that your program will run on the command line before submitting it. Programs that do not compile will receive greatly reduced credit.

Extra Credit: One option that will give you full credit is to re-implement A2LList<T> using recursion rather than iteration (this will require implementing the contains() and getNodeAt() methods recursively). Another option is to add one or more additional, non-trivial methods to either of your classes, making sure that they are entirely recursive. In either case, you must also write and submit a driver program that demonstrates the functionality of your recursive methods.