CS 204, Program #2: Linked Lists

Due Wednesday, Sept. 23rd by 11:59 p.m.

Summary: Implement the LinkedList code from the textbook and extend it to allow lists to be inserted into other lists.

The Problem: The textbook code for Linked Lists is written in the style of "pseudocode", intended to communicate a reasonably programming-language-free view of linked list code. You should copy this code into an Eclipse project (LinkedList class) and convert it into proper Java. In addition, you should convert it into proper Object-Oriented fashion, as described in class. You should adapt the textbook code for append, prepend, insertAfter, and search.

The main modification we'll be making to the Linked List code is to add the ability to insert one linked list into another one. Since we will be using multiple lists, this means we need to be programming with Objects, not just with a single class (static code) as we did with the first program. You also need to add the following methods to the LinkedList class: (i) A constructor that takes an array of Integers and converts it into a linked list of nodes with integers; (ii) An insertListAfterNode that takes a second list and a node in the first list (the "this" list) and inserts the entire list into the target list after "currentNode" and before currentNode–>next; (ii) An insertListAfterKey which takes a list and an integer key value, searches for that integer, then inserts the list after the node which contains that key.

Programming Requirements:

1) You must use the "Node" class posted on Moodle, with no changes. Your program should work correctly with the provided Driver class, although you are welcome to test your code with variations of that code.
2) You must have three class variables titled rodLength, leftTemperature, and rightTemperature. (rodLength is int; the others are floats), appropriately defined.
3) Except for the edits implied by the description above, you must use (variations of) the textbook code for the methods listed above. (It's easy to find dozens of versions of basic linked lists code online, but those aren't appropriate for this assignment.)
4) The LinkedList object must have "head" and "tail" as object variables, and no others.
5) No method should have more than 9 lines of code.
6) You must have the constructor that takes as input an array of integers and converts it into a Linked List. Do this by hand; do not use a Java library program to do this.
7) Your append and prepend methods must take a single node as an input, and return "void". Search must take as input an integer key value and return a Node (which is actually a pointer to the node).
8) insertListAfterKey must take the list to be inserted (as the 1st parameter), and the key (as the 2nd parameter) to look up a location in the linked list. It should not search for the key itself (which is an O(N) operation), but should instead call the search method to find the appropriate node and then the insertListAfterNode method. It should return void.
9) insertListAfterNode must take the list to be inserted and the Node as to where to insert it, in that order. This must be an O(1) (constant time) method.
10) toString( ) must return the string as indicated in the provided sample runs.
11) Your code must satisfy all of the provided JUnit tests (to be distributed soon).
12) Your output must meet all the formatting requirements of the Sample Runs.

Sample Run with the provided driver. There is no input, and only this test output. The “dummy” values “NNN” printed on the last two lines are number values that depend on the exact length of your code

```
Some of the first primes: 2, 3, 5, 7, 13
Added the missing 11: 2, 3, 5, 7, 11, 13
Added primes in the 30s: 2, 3, 5, 7, 11, 13, 31, 37
Added the missing primes: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37
The following should result in an exception halting the program:
Exception in thread "main" java.lang.IllegalArgumentException: Non-
existent key 21
	at linkedList.LinkedList.insertListAfterKey(LinkedList.java:NNN)
	at linkedList.Driver.main(Driver.java:NNN)
```