**CSCD 300 Homework Two**

Circular Doubly Linked List Basics and Advanced Operations.

Javadoc Standard Documentation \*NOT\* Required -- basic documentation is (all source files must include author name and a description of the class(es) contained in the file; any code another developer might not understand at first look should also be documented). Also, follow Java's naming conventions, utilize whitespace/indentation liberally in your code.

**Purpose**

The purpose of this assignment is to allow you to demonstrate basic knowledge of **Circular Doubly Linked Lists** and edge cases. These operations include (directly or indirectly): inserting, deleting, returning a sub-list, finding, modifying and "printing." This assignment should be done by you without outside help (other than from your instructor). **If you are unable to complete this assignment in the allotted time, it is a strong indication you are not ready for the material in this class and you MUST meet with the instructor to discuss your continued enrollment in the class.**

**Specifics**

* As defined in the CDoublyLinkedList class, you have to use a dummy node for your Circular Doubly Linked List.
* Please implement all the methods in the CDoublyLinkedList class except for the AddFirst() and toString() methods. The requirements and specifications for each method are provided in the comments above each method header. You are permitted to add helper methods into the class as needed.
* Please do NOT change the interface and signature (the list of arguments and their type, and return type) of the provided methods in the CDoublyLinkedList class.
* Please do NOT change the provided AddFirst() and toString() methods in the CDoublyLinkedList class.
* Please do NOT change any code in the provided Tester class. However, during your **testing and debugging**, you can comment out some method calls in the main() of the Tester class in order to single out the method that you are currently debugging. After finish debugging, please make sure all methods are called (uncommented) in the main() of the Tester class, as it was initially provided by the instructor.
* Please read and understand the design and implementation of the Tester class. So that you can imitate its design in other project when testing each method of your implementation.

**List of methods you have to implement**

1. void addLast(Object data)
2. CDoublyLinkedList subListOfSmallerValues(Comparable data)
3. boolean removeStartingAtBack(Object dataToRemove)
4. int lastIndexOf(Object o)
5. boolean retainAll(CDoublyLinkedList other)
6. void insertionSort()

**Rubrics:**

1. Implementation of methods 1 through 4 above weigh 10% each. And implementation of methods 5 and 6 above weigh 15% each. So it is 70% for all six method implementations.
2. A progress report for this assignment weights 30%. A progress report is a separate PDF file that will be due many days before the entire project is due. A separate assignment named **hw2Progress** will be created on canvas. Please turn in your progress report for this homework there. Please check the description of homework **hw2Progress on Canvas for more details.**

**EXTRA CREDIT (up to 10 points possible)**

You have to use the generic parameter E for the CDoublyLinkedList class and its Node class, as well as specified in the java documentation. You have to rewrite CDoublyLinkedList class so it supports generics (the stuff with the Es as type parameters). This means that when creating a CDoublyLinkedList you will specify the type of data it will hold (e. g. CDoublyLinkedList <String> myList = new CDoublyLinkedList <String>();) The previous code specifies your CDoublyLinkedList will only ever contain String objects as its data. The compiler will enforce this, which is a wonderful thing.

In this case, you can rewrite the Tester class to support the generic parameter E testing. But test results should be same as what were provided for the case where generic parameter E is not used.

**If you do this extra credit, be sure and clearly document it at the top of your CDoublyLinkedList class. Failure to do so will result in no extra points awarded. Your teacher will NOT help you with the extra credit portion of this homework.**

**To Turn In**

Turn in all your source code(.java file) on the **EWU Canvas** by going to CSCD300-01 course page on Canvas, then clicking Assignments🡪hw2->submit.

A progress report is a separate PDF file that will be due many days before the entire project is due. A separate assignment named **hw2Progress** will be created on canvas. Please turn in your progress report for this homework there. Please check the description of homework **hw2Progress on Canvas for more details.**

This assignment must be submitted in working order by the due data on the top of this assignment. Submit a zip file with your source files only. Source files are those that end in .java. Check your zip file before submitting and make sure it has only your source files. Do not submit .class files; they do the grader no good. You will not receive credit if your submission contains only .class files. Name your zip file with your last name, followed by the first initial of your first name, followed by hw2. For example, if you are John Smith, name you file as smithjhw2.zip.

The grader should be able to open your zip file, compile your code, and run your program from **the command line using commands, javac \*.java and java** **Tester** version 1.7 or greater (**so make sure you try this yourself before you submit**). **If your code has compile-time error when USING COMMAND LINE tool to compile, you get a ZERO for this homework! Therefore, please make sure you have NOT defined any java packages in your project, but have used the default java package.**

Get started right away on this assignment or you WILL NOT finish :-(

**CORRECT OUTPUT of the program is provided below.**

------------------testAddLast()----

{A}

{A->B}

{A->B->null}

{A->B->null->C}

-----------------------------

--------testSubListOfSmallerValues()----------

{}

{B->B->B->A}

{F->B->B->B->A->D}

{F->B->B->G->B->A->M->D}

-----------------------------

------------Test lastIndexOf()-----

-1

3

-1

-1

0

5

2

-----------------------------

---------testRetainAll()---------

{}

{6:Tony->6:Tony}

{null->bad->null}

-----------------------------

---------------Test removeStartingAtBack---

false

true

{apple->null->bad->null}

true

{apple->null->bad}

{2:Morning->3:Abby->4:Tim->5:Tom->6:Tony}

-----------------------------

---------test insertionSort()---------

{}

{D}

{D->E->E->F->G}

-----------------------------