CS 3305: Data Structures

Assignment 9 - AVL Trees (100 points total)

GENERAL SUBMISSION REQUIREMENTS

Upload all files individually as specified, not as zip files, to Assignments in D2L. Do not email files.
Make sure your program compiles, runs and produces the correct output.
Ensure you have the correct file name(s), and author header, as specified in the Assignment.
Always use meaningful labels for prompts, inputs, and outputs.
Always use comments, indentation and whitespace as shown in examples.
Note: Never hard-code test data in the test program, unless explicitly stated in the assignment.
Always allow the user to enter the test data using a menu option.

Assignment 09 – PART 1 Draw AVL Tree (50 points) :

Objective of this assignment is to reinforce understanding of AVL Trees. You will draw an AVL Tree on paper, given the following events:

Begin with an empty AVL Tree
Insert the following integers, in this order 8, 26, 49, 14, 1, 20, 35
Show the resulting tree **after each insertion**, and

1. make clear any rotations that must be performed,
2. show the resulting tree after that rotation

Note: This is a different format for the deliverable. You may draw your AVL Trees by hand on paper and then scan the images into a word document. Phones have great scanning apps. Save doc as a file named, LastName-A9-Part-1-DrawAVL.docx or .pdf and submit it to D2L.

Assignment 09 – PART 2 AVL Tree (50 points) :

Objective of this assignment is to reinforce understanding of AVL Trees. You may use the Java Libraries for solving this problem. No files or data are provided for this part of the assignment.
Please note, the following errors in the Text Book, listing 25.5 for BST.java:

(1. On Page 938, line 89, the method name preorder should be postorder. 2. On Page 938, line 102, the method name postorder should be preorder.)

Write a complete test program to test if the AVLTree class in Listing 26.3 on page 972 of the Liang textbook meets all requirements of an AVL Tree.

Note you have additional class dependencies as follows:

AVL extends BST, BST extends AbstractTree and AbstractTree implements

Tree You will need the following files

```
Tree.java        // listing 25.3
AbstractTree.java // listing 25.4
BST.java         // listing 25.5
```

Do not forget to include author header in each submitted file as shown, no header, no points!

```
// Name:         <your name>
// Class:        CS 3305/ put your section number after the /
// Term:         Spring 2023
/   Instructor:  Carla McManus
/   Assignment:  9-Part-2-AVL
```

Capture a **READABLE** screenshot(s) of your program output and paste into a word/pdf document. Readable means readable! Screenshots ***should not be an entire desktop*** – use some type of snipping tool. After your output screenshots, copy and paste the source code for your program into the word/pdf doc.

Make sure that the source you've copied and pasted into your word/pdf is the same as that in your .java source file that you will upload to D2L.

Save doc as a file named LastName-A9-Part-2-AVL.docx or .pdf. Last step is to upload word/pdf and .java files to D2L.

MAKE SURE YOUR CODE HAS COMMENTS ! We are getting submissions without comments in the code. No comments = (-20) points *per Part of the assignment*

Do not submit zip files.

Late penalties of 10 % per day are in effect for this assignment.