CS 3305/W03: Data Structures

Assignment 7 - Binary Trees Check D2L for Due Date (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 7 – Binary Trees (100 points) :**

Objective of this assignment is to reinforce understanding of Binary Trees and recursion.

Write a program to implement Binary Trees. You may use the Node class from the Java Library, or write your own Binary Tree Class, and implement the methods listed below. Hint, review section 25.2 (starts on p. 930) in the text for designing and implenting binary trees using linked lists.

Data is provided for this assignment. The data is in the form of sample tree arrays, listed in the additional data file that is attached to the Assignment as a file called include-A7-BT-testdata.txt.

**You are NOT going to read this data from the** file into your program !

You will hard code the test data that is provided in include-A8-BT-testdata.txt, i.e., copy and paste the data that represents different sample trees. In addition, note that the sample trees use pre order and in order traversals, as indicated by the sample data array names. For example, the array named, double pre[ ] is data used for pre-order traversal, and the array named double in[ ] is for in-order traversal. After the insertion of data for each tree, you must determine whether the tree is a binary search tree or not, and provide that information in your test results.

HINT: Create one tree at a time, using the data provided for that tree, execute the methods on that tree, write the results to the screen, and get a screenshot of each output.

**METHODS**

1. depth

if root is NULL, return -1

otherwise, determine maximum of the depths of the left and right subtrees, add 1 and return

2. max

test that root is not NULL

get the max in the left subtree, the max in the right subtree, and the data value in root

return the largest of these three values

ignore a subtree if it is empty

3. tree\_sum

returns the sum of the values in all the nodes in the tree

4. tree\_average

returns the average of the values in all the nodes in the tree

5. tree\_is\_balanced

returns true if and only if the tree is balanced

this means that the absolute value of the difference in depth between the

two subtrees of root is no more than 1.

it also means that both the left and the right subtree are balanced.

An empty tree is considered to be balanced.

if root is empty, the tree is balanced

otherwise, if either the right or the left subtree is not balanced,

then the tree is not balanced

otherwise, if the difference in depths between the left and right subtrees

is greater than one then the tree is not balanced

otherwise, the tree is balanced

TEST: To test the above methods, create 6 trees, one at a time, using the sample data and write your results in the listed form shown below

Tree\_Name

Binary search tree ? Yes or no // indicate whether the tree is a BST

Depth

Max

Sum

Average

Is Balanced ?

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: Semester Year  
// Instructor: Sharon Perry  
// Assignment: 7-BTs

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. Save doc as a file named LastName-A7-BTs.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 = ( -30 ) points *per Part of the assignment*

Do not submit zip files.

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

**Assignment 7 – Binary Trees – EXTRA CREDIT (20 points)**

Redo Assignment 7 using Linked Lists (if you previously used arrays/array lists) OR redo Assignment 7 using Arrays/Array Lists (if you previously used Linked Lists).

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: Semester Year  
// Instructor: Sharon Perry  
// Assignment: 7-BTs- EXTRA CREDIT

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. Save doc as a file named LastName-A7-BTs-ExtraCredit.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 = ( -30 ) points *per Part of the assignment*

Do not submit zip files.

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