CURTIN UNIVERSITY (CRICOS number: 00301J)

Department of Computing, Faculty of Engineering and Science

Data Structures and Algorithms (COMP1002)

PRACTICAL 4 –TREES

AIMS

- To implement a Binary Tree.
- To traverse a tree
- To add file IO and serialization to the tree

BEFORE THE PRACTICAL:

- Read this practical sheet fully before starting.
- Make sure you have completed Practical 3.

ACTIVITY 1: CREATING TREENODE

Use the lecture notes create a DSATreeNode class - this has already been written for you but you'll need to understand and test it.

ACTIVITY 2: CREATE BINARY SEARCH TREE

Use the template from the lecture notes to implement a Binary Search Tree - DSABinarySearchTree. Follow the pseudocode from the lectures to guide you here. The code for find() was already implemented for you. The methods must all use the **recursive** approaches and pseudocode from the lecture slides.

As you work, write a test harness to test each operation thoroughly. You may want to leave "delete" until you finish the rest of the practical and then come back to it. You can use the data from RandomNames7000.csv or use your own, but be sure to test all the cases.

ACTIVITY 3: ADD CODE TO TRAVERSE THE TREE

The lecture notes described the approach for doing inorder, preorder and postorder traversals of a tree. Add recursive implementations of these algorithms to output the traversed tree. You may want to export it as a queue...

ACTIVITY 4: IMPLEMENT ADDITIONAL OPERATIONS

The lecture notes described the approach for doing min, max and height. Implement and test each of these operations. Now consider how you would give a percentage score for how balanced the tree is. Implement this approach as a new method called "balance".

ACTIVITY 5: ADD FILE I/O TO YOUR OVERALL PROGRAM

Set up a menu system provide options including:

1) read a CSV file
2) read a serialized file
3) display the tree
4) write a CSV file - ask if they want inorder, preorder or postorder traversal
5) write a serialized file

Add the code required for these tasks with files of integers. Using these routines you should be able to test your traversals and explore re-reading in those files to create new trees.

SUBMISSION DELIVERABLE:

Your completed DSABinarySearchTree and related files/classes are due before the beginning of your next tutorial.

SUBMIT ELECTRONICALLY VIA BLACKBOARD, under the *Assessments* section.

MARKING GUIDE

Your submission will be marked as follows:

- [2] UML for all implemented classes. Include the methods (excluding basic getters and setters).
- [2] Your DSABinarySearchTree and DSATreeNode are implemented properly. You have test harness code to demonstrate it working.
- [2] You have implemented and can demonstrate the traversal code.
- [2] You have implemented an tested methods for min, max, height and balance.
- [2] You have implemented and can demonstrate the file I/O code.