Data Structures & Algorithms

Lab Exercise 5– Binary Tree Project (Part 1)

Due: to be demonstrated by week 17 (wk beginning 21-Jan)

Learning outcomes
At the end of this lab you should be able to:

- Create an instance of LinkedBinaryTree and populate it with nodes.
- Describe the structure of a binary search tree (BST)
- Write recurive and non-recursive implementions of the depth algorithm
- Demonstrate usage of the BinaryTreeIterator class
- Explain the difference between a tree node (BinaryTreeNode) and a tree iterator (BinaryTreeIterator)

Q1. The source files in the binary tree project are an implementation of a linked binary tree structure with the following inheritance hierarchy:

```
Container
     |
SimpleTree
     |
BinaryTree
     |
LinkedBinaryTree
```

(i) Review the recursive depth algorithm for a binary tree (data structures lecture 19, slide 17 onwards). Write an implementation of depth for the `LinkedBinaryTree` class.

1) Start by adding a pure virtual function to the `SimpleTree` interface.
2) Now write the implementation in the `LinkedBinaryTree` class.
3) Open the source file with the `main()` function and test your implementation of depth on the binary tree I have defined.

(ii) Now write a non-recursive version of the depth algorithm.

Q2. Consider a new member function for class `LinkedBinaryTree:`

```
Iterator treeMinimum(Iterator & n) const;
```

This algorithm should find the minimum value for the subtree that is rooted at a given node. Parameter `n` is the root of this subtree and the return value should be an iterator that points to the minimum value of this subtree. Write an appropriate implementation of this function.