Assignment 6

Due: April 21, 2010, no later than 3:00pm

Objective: Binary trees are among the simplest and most intuitive data structures studied in undergraduate Data Structure and Algorithms courses. And with good reason: binary trees have many interesting properties that recommend them as the data structure of choice in almost all areas of computer science and engineering. One of the side benefits of their simple, yet rich structure, is that binary trees can be implemented easily and, in some special cases, without the use of pointers, an important consideration in early programming languages such as, for example, FORTRAN and BASIC that lacked pointers.

And, of course, binary trees have been specialized to meet the specific needs of various applications. For example, full binary trees that are the object of this assignment, find applications to expression evaluation in parallel processing and compiler design. While one can look at full binary trees from many angles, this assignment takes the view that they are recursively defined objects. As it turns out, this view simplifies considerably the task of reasoning about full binary trees.

Please be reminded that the assignment is strictly personal and giving/receiving undue help in solving the problems is a violation of the Honor Code of Old Dominion University and shall be dealt with accordingly. If you are not sure what constitutes a violation of the Honor Code please do not hesitate to ask your instructor or the Old Dominion University Honor Council.

Statement of your assignment:

The class of full binary trees (FBT, for short) is defined recursively as follows:

B: The empty binary tree is an FBT

I: If $T_1$ and $T_2$ are either both empty FBTs, or both non-empty FBTs, then the tree obtained by adding a new root and making, respectively, $T_1$ and $T_2$ into the left and right subtrees of the new root is an FBT.

C: Nothing is an FBT unless it was obtained by one of the previous clauses.
As an illustration, the binary tree depicted in Figure 1 is an FBT.

![Binary Tree Image]

Figure 1: Illustrating a full binary tree.

Problem 1. (25%) Show the complete sequence of operations that, starting with an empty FBT, constructs the FBT in Figure 1. Show all the partial FBTs involved.

Problem 2. (25%) Prove by induction that a node of an FBT is either a leaf or else has exactly two children.

Problem 3. (25%) This is the converse of Problem 2. Prove that a binary tree with the property that every node is a leaf or else has exactly two children is an FBT.¹

Problem 4. (25%) Prove by induction that every non-empty FBT contains an odd number of nodes.

¹ For all practical purposes, this establishes an alternate, equivalent, definition of FBTs based on the regularity spelled out above. It is precisely this property that suggested the name of the class itself.