CS 251
Spring 2009
Voicu Popescu
Assignment 5
Due Friday February 20, 11:59PM

Binary trees

1. Implement a C++ binary tree class **BinaryTree** that models arithmetic expressions. An internal node stores an operator represented with a character (i.e. '+', '-', '*', '/') and a leaf stores an integer. It is OK to provide space for both and derive the type of node from the presence or absence of children. Provide the following functionality:
    1. A **constructor** that initializes and builds the binary tree from an arithmetic expression read in from the standard input.
        i. The expression has parentheses isolating all (<operand0> <binary operator> <operand1>) expressions, where <operandi> can be an expression, recursively. For example 1+2+3 is illegal input that you do not have to worry about. The legal form is ((1+2)+3).
        ii. All numbers are integers between 0 and 9.
        iii. Caution: handle space characters correctly. The string defining the expression could or could not have spaces: ((1 + 2)+3) should work just as well as ((1+2)+3)).
        iv. Caution: handle negative numbers correctly: (1*-5) is legal input.
        v. You do not have to worry about illegal input.
    2. A **destructor** that frees all dynamically allocated memory.
    3. A method **evaluate** that returns the value of the expression stored in the tree.
    4. A method **draw** that visualizes the binary tree in a text file called `tree.txt', with characters as pixels.
        i. An internal node should be drawn with a 5x5 pixels square, with the operator at the center. The border of the square should be one character thick, and should be represented with the character 'I'.
        ii. A leaf should be drawn with a 5x5 pixels square, with the number at the center. Use 'E' for the border.
        iii. A node should be drawn midway between its two children; place the square close to left child if you encounter half pixel issue. (See provided examples)
        iv. All nodes with the same depth should appear on the same row
        v. Extra credit (2%): Draw a link as a line segment connecting node centers; the line should stop before entering the node square.

2. Extra credit (2%): Add the member function **isBinaryTree(BinaryTree *bTree)** to BinaryTree that verifies that a tree of nodes given by a pointer to its root is indeed a binary tree. The properties that you have to verify are:
    1. All nodes have 0 or 2 children.
    2. A node is the parent of its children and of no other node.
3. Turn in instructions
    1. Question 1: Name your program file as "binaryTree.cpp". "binaryTree.cpp" should implement the operations given in "binaryTree.h". You may add some code to "binaryTree.h", in fact you should. However, do not change the names and signatures

of the given operations. A test file “binaryTreeTest.cpp” is also provided to test the data structure. Examine and run that program to test your operations. You will not turnin the test file.

2. Turnin a blank file named “extracredit” for your extra credits to be graded (do not turnin this file if you did not attempt at least one of the extra credit questions). Create this file in the A5 directory with the following command
   touch extracredit

   Please provide the following public methods in you BinaryTree class if you are attempting extra credit Q2. Each method should return a pointer to a BinaryTree object or null if it does not exist.
   BinaryTree*& getRefParent();
   BinaryTree*& getRefLeft();
   BinaryTree*& getRefRight();

3. **Read General turnin instructions parts E, F and G very carefully, on the website.**

4. Your codes MUST compile on lore.cs.purdue.edu for it to be graded.

5. All of your documents (binaryTree.h, binaryTree.cpp, extracredit (OPTIONAL)) MUST be in the directory A5.

In the directory above A5, run the command on lore.cs.purdue.edu

```
turnin -c cs251 -p A5 A5
```

Optionally run the following command to verify what files were submitted

```
turnin -c cs251 -p A5 –v
```

Warning: turnin will be automatically disabled at 02/20/2009 at 11:59 PM EST and you will be unable to turnin your assignment after this time.

Warning: turning in multiple times is ok but only the last one will be graded as each turnin permanently overwrites the previous one.

Warning: failure to turnin exactly according to the instructions given above will result in your A5 receiving a grade of 0.