CS 11 Data Structures and Algorithms

Assignment 13: Binary Search Trees 1

Skip to Main Content

Assignment 13.1

Don't start this assignment yet. I will be completely rewriting it. If you are reading this after April 24, please let me know.

No documentation is required on this assignment.

Start with the binaryTree class provided in lesson 23 and make the following changes. Don't make any changes other than the additions listed here. (Adding private helper functions is also ok.)

1. **mSize**: Add a data member to store the size of the tree. Call it "mSize" to distinguish it from the "size()" function. You'll need to update this new data member in all of the appropriate places in the class implementation. Change the size() function so that it returns this data member instead of calculating the size.

2. **numPrimes()**: Add a "numPrimes()" function to the class that returns the number of nodes in the tree that contain prime integers. Good decomposition dictates that you will want a helper function that determines whether its parameter is prime. Don't worry about making this efficient. Just test all of the numbers less than the parameter and if any of them divide evenly into the parameter, then it's not prime.

3. **toLL()**: Add a "toLL()" function to the class that converts the calling binary tree object into an LL object. The items in the LL object should be in increasing order. The created LL object should be returned. Here's a sample client to illustrate how this function might be used:

```
#include <iostream>
#include "LL.h"
#include "binarytree.h"
using namespace std;

int main() {
    binarytree t;
    for (int i = 0; i < 20; i++) {
        t.insert(rand() % 50);
    }

    cout << "The binary tree: ";
    t.print();
    cout << endl;

    LL<int> l;
    l = t.toLL();

    cout << "The linked list: ";

    for (LL<int>::iterator i = l.begin(); i != l.end(); i++) {
        cout << *i << " ";
    }
    cout << endl;

    cout << endl;
}
```

4. **Big-3**: Finally, add the big-3 to the class. I would recommend that you first add private functions named "copy()" and "clear()". Each of these will have an "aux" friend function. If you write these correctly, your big-3 will be trivial.

Submit Your Work

Use the Assignment Submission link to submit your binarytree.h and binarytree.cpp files. No client file or output is required. When you submit your assignment there will be a text field in which you can add a note to me (called a "comment", but don't confuse it with a C++ comment). In this "comments" section of the submission page let me know whether the class works as required.