FINAL PROJECT

Much Ado About Trees.

Due Thursday, May the 15th, 11:59pm

This assignment is a ***project***, *not* a homework. It is ***not*** subject to the “lowest grade dropped” policy. For your final grade, this project is worth *twice* the value of a homework.

You need to submit a file called ‘`MyBSTree.h`’ and any other necessary implementation files. Remember to put your name and section at the top of all your files.

Submit using ‘f’ for the assignment number, as in: “ `cssubmit 153 <section> f` ”

Problem

Professor Farnsworth is on the verge of a breakthrough in genetic engineering: Trees!. But not just any trees, but “any-trees”. Apple-trees, Orange-Trees, Pizza-trees, Money-trees, Integer-trees. In other words, trees that grow anything that you genetically engineer them for.

A fruit-salad tree

These “any-trees” branch out with a factor of 2 (they are Binary Trees) and maintain a peculiar order among their branches: (The Search Property!)

Your job is to write a program that simulates the behavior of these trees. Knowing that these are going to be Binary Search Trees, be ready with your recursive tools.

Testing

Use the provided tester file to check if your implementation is working correctly.

- The program ‘treetester.cpp’ uses the ‘MyBSTree’ class and the intended output is ‘treeoutput.txt’.

Notice:

1. You are expected to derive the ‘AbstractBSTree’ class.
2. You are also expected to implement the “Big-3” for your ‘MyBSTree’ class
3. The functions ‘findMin()’ and ‘findMax()’ are expected to “throw” errors.
4. ‘contains()’ is **<u>not boolean</u>**, read the description carefully, and see the example.

Useful Hints

1. Carefully read the comments of each member function.
2. Write down an algorithm for the function before you start coding it.
3. Develop your member functions one at a time, starting from the simplest ones.
   Move to the next function only after the previous one has been tested.
   Trying to code the whole class and then remove the bugs may prove to be too big a task.
4. Use the test functions one at a time.

More Hints

- Use <u>2 classes</u>, One for the tree nodes, and another as an encapsulation class. That way, you can more easily use recursion and use the NULL pointer as your base case.
- In order to use member functions and recursive functions, have the member function call the recursive version, as in the example.

Example:

```
struct TreeNode
{
   T m_data;
   TreeNode* m_right;
   TreeNode* m_left;
}

class MyBSTree
{
   TreeNode* m_root;
   int m_size;

   int recursive_foo( TreeNode& t1 )
   {
      … …
      recursive_foo( t2 );
      … …
   }

   int foo()
   {
      recursive_foo( m_root );
   }
}// class MyBSTree
```

HINT: Printing a Tree

The following is the code for “Pretty Printing” a tree:

```
template <typename T>
void prettyPrint (const TreeNode<T>* t, int pad)
{
  string s(pad, ' ');
  if (t == NULL)
      cout << endl;
  else{
    prettyPrint(t->right, pad+4);
    cout << s << t->data << endl;
    prettyPrint(t->left, pad+4);
  }
}
```

HINT: Cloning a tree

Check this slick recursive function to clone a tree

```
template <typename T>
TreeNode<T>* clone(const TreeNode<T>* t)
{
  if (t == NULL)
      return NULL;
  else{
      return new TreeNode<T>(t->data, clone(t->left),clone(t->right));
  }
}
```