New Beginnings – Summer 2018

C++ Programming - Linked List Practice – Part IV

In this part, we'll migrate our Linked List to a BST – Binary Search Tree. This means replacing the List class with a Tree class and the Node class with a Leaf class.

There are a couple of ways to approach building a tree, but for this practice, I'm providing the class interfaces. You'll need to implement the member functions.

Leaf class

```
class Leaf {
        private:
        StudentRecord * data;
        int id;
        Leaf * left;
        Leaf * right;

        public:
        Leaf();
        ~Leaf();
        void setLeft(Leaf *);
        void setRight(Leaf *);
        Leaf * getRight();
        Leaf * getLeft();
        void setId(int);
        int getId();
        void setData(StudentRecord *);
        StudentRecord * getData();
        void display();
};

class BinarySearchTree {
        private:
        Leaf * root;
        void deleteTree(Leaf *);
        public:
        Tree();
        ~Tree();
        void addStudent(int, StudentRec *, Leaf *); // Inserts student with int id and StudentRec
        StudentRec * findStudentById (int, Leaf *); // Finds the student with int id
        void display(Leaf *);
}
```

1. The findStudentById() function is recursive – looking for a Leaf with int id starting at the given Leaf. For example:
    a. tree->findStudentById(42, tree->root);
2. You'll want to spend some time thinking about the addStudent method – specifically how to traverse down the tree to the bottom of the appropriate branch.
3. Likewise, think about the Tree destructor. The deleteTree(Leaf *); method will come in handy to delete the tree recursively.
4. The display function for Tree is also recursive.

For all of the recursive functions, take some time to think about why we need to pass in the Leaf pointer?

DISCLAIMER There may be some additional class member functions needed. (I haven't implemented this completely, so I may have forgotten something!)