CS 2123 Data Structures

Programming Project 4

In this project, we will be implementing the basic functionality of a Binary Search Tree.

Files provided in the project:

1) BinarySearchTree.h. This file contains the structs needed for the BinarySearchTree as well as the prototypes for several functions that you will need to implement. In particular you need to implement the following functions in the file BinarySearchTree.c:
    a. **BinarySearchTree newBinarySearchTree()** which should allocate the memory for a new binary search tree, initialize the variables, and return the address.
    b. **void freeBinarySearchTree(BinarySearchTree tree)** which should free the tree (including any nodes currently in the tree).
    c. **NodeT *allocateNode(Element value)** which should allocate memory for a new node, store "value" inside this node, and return the address of the node.
    d. **NodeT *search(NodeT *p, int searchValue)** which should **recursively** search the subtree rooted at p for a node containing a key value equal to searchValue. If such a node exists, return a pointer to the node. If no such node exists, return NULL. Note that to search the entire tree, we would call this function with search(tree->pRoot, searchValue).
    e. **int insert(BinarySearchTree tree, Element value)** which will insert a node with the element value into the tree as a leaf node if it does not already exist in the tree. If the node is inserted then return true. If the node already exists, return false.
    f. **void printInOrder(NodeT *p)** which will recursively print the values of the nodes in the tree in increasing order. We would call this function with printInOrder(tree->pRoot).
    g. **void printPreOrder(NodeT *p)** which will recursively print the values of the nodes in the tree according to a preorder traversal (discussed in class on Thursday). We would call this function with printPreOrder(tree->pRoot).

2) p4Input.txt. This file contains a sample input file that you should process. Each line will contain either INSERT X, SEARCH X, PRINT INORDER, or PRINT PREORDER. If the line contains INSERT X, then you should insert a new node into the binary search tree containing the value X.

If the value inserted correctly, print “Inserted X into tree”. If the node was already in the tree, print “X already is in the tree”. If the line contains SEARCH X, you should search the tree for a node containing X. Print “Found X” if it exists and print “X not in tree” if it does not exist. If the line contains PRINT INORDER or PRINT PREORDER then print the contents of the tree with an inorder traversal or a preorder traversal respectively.

3) abc123p4.c. Rename this file to your abc123. This file is mostly empty right now. It contains the main() function that you should complete. In main, you should create a new BinarySearchTree, read a line from p4Input.txt (you can do this however you would like), and perform the appropriate operation on the tree.

5) Makefile. Update the makefile to reflect your abc123. Compile using **make p4**. Execute the program using **./p4 < p4Input.txt**

Submitting:

Please submit to the blackboard dropbox a zipped folder containing each of the files we have provided you along with the new DoublyLinkedList.c and BrowserList.c files that you create.