2413 Assignment 7

Due Date
3-21-16 11:59pm

For this assignment you will be implementing a Binary Search Tree (BSTree) which contains integers. I highly recommend writing your functionality in a recursive way. Duplicate integers are allowed in the tree

The Binary Search Property Applies to every node :

1. The elements of the left subtree are $\leq$ the current node's element
2. The elements of the right subtree are $>$ the curren node's element

Input Commands From Standard In:

```
insert <integer>
remove <integer>
postorder
preorder
inorder
calculate
clear
```

1 insert $< integer >$

Given an existing BSTree, this operation will add the integer as a leaf node to the tree. Different orderings of the insertion command lead to different tree structures.

Example 1:

```
insert 5
insert 10
```

5 is inserted as a leaf into the empty tree and 10 is inserted as a leaf in the right sub-tree of 5.

Example 2:

```
insert 10
insert 5
insert 6
```

10 is inserted as a leaf into the empty tree. 5 is inserted as a leaf in the left subtree of 10. 6 is inserted as a lead in the left subtree of 10 and the right subtree of 5.

2 remove $< integer >$

Given an existing BSTree, this operation will remove all instances of the integer. Your code should have the following behavior: For a given node in the tree, first remove the integer from the left sub tree. Then check if the current node matches the integer. If it does, insert the root of the right subtree into the proper open leaf position of the left subtree.

If tree is empty, print: "Tree Is Empty" on its own line
Example 3:

```
insert 10
insert 5
insert 11
remove 10
```

Before the remove happens, 10 is the root, 5 is its left child, 11 is its right child. After the remove, 5 is the root and 11 is its right child.

3 postorder

Print the integers of the tree followed by a single white space in post-order traversal. When done, print a newline character.

1. print left subtree
2. print right subtree
3. print current node

If tree is empty, print: "Tree Is Empty" on its own line

4 preorder

Print the integers of the tree followed by a single white space in pre-order traversal. When done, print a newline character.

1. print current node
2. print left subtree
3. print right subtree

If tree is empty, print: "Tree Is Empty" on its own line

5 inorder

Print the integers of the tree followed by a single white space in in-order traversal. When done, print a newline character.

1. print left subtree
2. print current node
3. print right subtree

If tree is empty, print: "Tree Is Empty" on its own line

6 calculate

Call reccalculate on the tree's root and print the returned result on its own line.

Given a node, reccalculate is performed as follows:

1. if the node is NULL, return 0;
2. if the node has no children, return the node's value.
3. A is the result of recursively calling calculate on the root of the left sub-tree.
4. B is the result of recursively calling calculate on the root of the right sub-tree.
5. return (current node's value) * (A-B)

If tree is empty, print: "Tree Is Empty" on its own line

7 clear

Deletes all the nodes in the tree, leaving it as an empty tree.

8 Input Output Pairs

8.1 Case 1

INPUT:

```
remove 5
inorder
postorder
preorder
calculate
clear
insert 6
calculate
insert 7
calculate
insert 5
calculate
```

OUTPUT:

```
Tree Is Empty
Tree Is Empty
Tree Is Empty
Tree Is Empty
Tree Is Empty
0
-42
-12
```

8.2 Case 1

INPUT:

```
insert 10
insert 20
insert 17
insert 21
insert 5
```

```
calculate
remove 20
calculate
```

OUTPUT:

```
850
3620
```