Test server output the following.

EST #4
  timeout 1s ..:/vbst-test/littletest-vbst

Testing these operations:
  vbst *a = newVBST(displayInteger,compareIntegers);
  insertVBST(a,newInteger(13));
  insertVBST(a,newInteger(13));
  displayVBST(fp,a);

Your output:
0: 13(13)-
1: =13(13)-r
Expected output:
0: =13-2(13-2)-

Output does not match.

The test did not succeed

TEST #5
  timeout 1s ..:/rbt-test/littletest-rbt

Testing these operations:
  rbt *a = newRBT(displayInteger,compareIntegers);
  insertRBT(a,newInteger(13));
  insertRBT(a,newInteger(13));
  displayRBT(fp,a);

Your output:
0: 13-B(13-B)-
1: =13-R(13-B)-r
Expected output:
0: =13-2-B(13-2-B)-

Output does not match.

The test did not succeed

Locally the same tests yield the following..
EST #4
Testing these operations:

```c
vbst *a = newVBST(displayInteger,compareIntegers);
insertVBST(a,newInteger(13));
insertVBST(a,newInteger(13));
displayVBST(fp,a);
```

Your output:
0: =13-2(13-2)-
Expected output:
0: =13-2(13-2)-

TEST #5

timeout 1s ../rbt-test/littletest-rbt

Testing these operations:
```c
rbt *a = newRBT(displayInteger,compareIntegers);
insertRBT(a,newInteger(13));
insertRBT(a,newInteger(13));
displayRBT(fp,a);
```

Your output:
0: =13-2-B(13-2-B)-
Expected output:
0: =13-2-B(13-2-B)-

Seems to work fine locally.. Thoughts?

---

Subject: Re: Difference of output from test2 server
Posted by bmbaker1 on Thu, 09 Mar 2017 21:16:14 GMT
View Forum Message <-> Reply to Message

My first thought is that you are doing something in bst.c that should be done in rbt.c or vbst.c. Possibly the frequency count. Sorry if that's probably not that much help.

---

Subject: Re: Difference of output from test2 server
Posted by djkoelz on Thu, 09 Mar 2017 21:46:29 GMT
View Forum Message <-> Reply to Message

For simplicity lets just look at the vbst case

// vbst.c
Subject: Re: Difference of output from test2 server  
Posted by djkoelz on Thu, 09 Mar 2017 22:41:22 GMT  
View Forum Message <> Reply to Message

I have a suspicion that the bst module being substituted does the following... I will take liberties and leave out details where not important

```c
 insertVBST(vbst* tree, void* value)  
 {  
    vbstValue* nodeValue = newVBSTValue(value, ...)  
    bstNode* node = insertBST(tree->tree, nodeValue)  
    ((vbstValue*)node->value)->frequency++;  
 }

// bst.c

insertBST(bst* tree, void* value)  
{  
    // not sure how much I can actually say here that is still within the rules of this forum  
    // but general idea... iterate through the tree (right/left in accordance to compareValue returns)  
    // if you get to a null create the new node there  
    // if the values are equal return the equal node  
 }
```

```c
 Subject: Re: Difference of output from test2 server  
Posted by djkoelz on Thu, 09 Mar 2017 22:41:22 GMT  
View Forum Message <> Reply to Message

I have a suspicion that the bst module being substituted does the following... I will take liberties and leave out details where not important

```c
 insertBST(bst* tree, void* value)  
{  
    bstNode* node = tree->root;  

    while(1)  
    {  
        if (!node)  
        {  
            // create new node  
            // return newly created node  
        }  

        if (tree->compare(node->value, value) > 0)  
        {  
            node = node->left;  
            continue;  
        }  
        else // this is the culprit.. this should be else if (tree->compare(node->value, value) < 0)  
        {  
            node = node->right  
        }  
```
There are three cases when testing validity of node equality. It seems that the bst module being substituted does not account for all three cases.

Suspicions seem confirmed.

By changing the insertion methods for rbt and vbst as so the correct functionality seems to be performed.

```c
// vbst

insertVBST(vbst* tree, void *value)
{
  // create new node value
  // search for the new node value
  // if it doesn't exist, create and insert the new node
  // increase frequency of node
}
```

This is silly... are people doing this?
We effectively search for the node twice.

This would be solved by fixing the insertBST function to account for equal valued expression to return the node currently being evaluated.

Perhaps I am wrong though.

I would like to know though, for SCIENCE!