Concerning the following functions:

```c
int findBST(bst *tree, void *value); //returns 1 if found, 0 otherwise
bstNode *findBSTNode(bst *tree, void *value); //returns 0 if not found
```

Are these essentially the same operation, only with different return types? If so, wouldn't (int) findBST be redundant, since you could just run findBSTNode and test for a NULL value?

Also, it seems like it would be a good strategy then to write findBSTNode first, then implement findBST as a call to findBSTNode that returns 0 if findBSTNode returns a NULL value and 1 otherwise.

Am I wrong?

---

Subject: Re: Find BST
Posted by davidmccoy on Sun, 26 Feb 2017 23:27:43 GMT

It looks like they would both use a helper function that found a node with the desired value (or null if not found), and then for findBST return an 1 if a node was found or 0 if not, and for findBSTNode just return the node pointer found.

---

Subject: Re: Find BST
Posted by nltollman on Mon, 27 Feb 2017 05:11:40 GMT

I don’t see the point of having the parameter of findBSTNode to be the BST itself rather than a BSTNode. If you have the parameter as BST, you would need to create a private function that’s recursively called to traverse down the tree and find the node. However, if you have the parameter as a BSTNode, you wouldn’t need a private function at all. You would simply have that function itself, findBSTNode, become a recursive function.

---

Subject: Re: Find BST
Posted by lusth on Mon, 27 Feb 2017 20:26:00 GMT

findBSTNode is meant for the application (vbst and rbt) and those applications won't know the internals of a bst (such was what the root pointer is called).
cdyancey wrote on Sat, 25 February 2017 16:11

Concerning the following functions:

```
int findBST(bst *tree, void *value); //returns 1 if found, 0 otherwise
bstNode *findBSTNode(bst *tree, void *value); //returns 0 if not found
```

Are these essentially the same operation, only with different return types? If so, wouldn't (int) findBST be redundant, since you could just run findBSTNode and test for a NULL value?

Also, it seems like it would be a good strategy then to write findBSTNode first, then implement findBST as a call to findBSTNode that returns 0 if findBSTNode returns a NULL value and 1 otherwise.

Am I wrong?

I noticed Dr. Lusth didn't try to correct this logic, so I'm assuming this is how it's supposed to go?
It DOES NOT function as a binary/boolean function of found or not found.

Project Description under INTRODUCTION section states:

"You will compare the performance of vanilla binary search trees versus red-black trees by reading in a corpus of text, storing the word and phrases therein into a search tree, and then performing operations on the resulting tree. The corpus will be stored in a file; the commands for manipulating the tree will be stored in a file as well. You will implement the following operations for both kinds of trees:
- insert which inserts a value into the tree (or updates a frequency count)
- delete which decreases a frequency count (or removes a value from the tree)
- find which reports the frequency of a value in the tree"

Lusth please respond to confirm!

Subject: Re: Find BST
Posted by bmbaker1 on Wed, 08 Mar 2017 17:38:48 GMT

bmbaker1
I do not believe there is anyway to return the frequency of a node's phrase from bst.c without casting it to a vbst or rbt object first. I'm pretty sure that we are not supposed to do that. That being said, per my other reply in this thread, I never used findBST(), so...

Subject: Re: Find BST
Posted by bkaaron on Wed, 08 Mar 2017 18:04:11 GMT

bkaaron
If what you say is true, then we need a specs change. I quoted directly from the specs. It tells us what find is supposed to do. We need a response from Lusth before the official due date, but definitely before the grace period ends. This function, if tested as it is described in the spec, will fail out for anyone who returns boolean notation only.

Subject: Re: Find BST
Posted by bkaaron on Wed, 08 Mar 2017 18:11:48 GMT

bkaaron
I see my mistake. In regard to the vanilla findBST the spec is not incorrect if you return Boolean, because you handle the frequency return in the inheriting function's findRBT/findVBST respectively. This is my mistake. As long as those functions return non-boolean values you are fine. findBST need not return anything other than boolean 0/1.
You are correct Baker.

Subject: Re: Find BST
Posted by lusth on Wed, 08 Mar 2017 19:21:01 GMT

Glad *that's* settled.