I am testing my normal BST class, which is exhibiting the following abnormal behavior:

Every time I insert a new node, every node in the tree is replaced with the value I'm trying to insert.

The only time I change a node's value is when I create a new node in the insertBST function. Similarly, I set the root of the tree exactly once and I have checked to make sure I am not performing operations on different trees. The insert function follows the pseudocode in the book which has been verified as working for several students.

The program still compares the value of different node pointers as it "walks down" the tree, but all of those nodes take on the value of my new node. The comparator function is working properly, but always returns 0 since it is being passed in two identical values.

The weird thing is, I can check the root of the tree from my main class before I perform my insertBST operation. After one insertion, things are fine. For the second insertion, in the BST class, I check the root again as soon as the insertBST function is called - and it is the value of the node I'm trying to insert. The root has changed somehow while doing nothing but passing the same tree and a new value (that hasn't even been inserted yet) to my BST class.

No operations are performed on the tree outside of the BST class. (except calls to newBST and insertBST from the main class for testing) The tree is not a static variable and I cannot find any scope issues that would let any class other than bst.c to interfere with the tree.

I have gone over every single line in my code several times and cannot figure out what's going on here. Because I can't determine why or how the issue occurs, I'd have to have someone vet my entire code - which obviously I can't post on the forums.

I have been blocked on this seemingly trivial error for three days now, and I'm out of ideas other than it being an environment/memory allocation issue. (I am working within a Xubuntu virtual machine)

PM me if you would like to see portions of my code.

because I know everyone has better things to do than to debug my code If anyone has any clue of what could be going on here, I'll buy you donuts before class. seriously

.
I bet you are using strings to insert, and if you changed your value to integers, your code would work fine. The issue is how you are allocating/assigning the pointers when reading from a file then inserting it. All your pointers now point to the new value you are inserting and that's why everything changes to the new value. Same thing happened to me. I think I used a string copy function to get around this.

---

On the subject of absurd problems with code, I am having trouble with a similar issue. When I create a bst object using newVBST(), it allocates space, initializes value and so on. On of the things it does is set the value of root to NULL. Immediately after returning that bst object to the bst* in vbst, the root value is no longer NULL. I fixed this by setting root to NULL again even though it should not have changed and that fixed the issue.

Now I have the same problem but with function pointers to display and compare in vbst. newVBST() sets up its display and compare functions just fine but when it returns the vbst object it created to bstrees.c, bstrees.c sees the compare function as being NULL. This causes a seg fault whenever tree->compare is called in bst.c. I am on Mac and it does not work on my system but I had someone run it successfully on it on Ubuntu VM where it worked so it seems to be an environmental issue. Maybe try running yours on a different system.

---

catwater wrote on Sun, 05 March 2017 13:25 I bet you are using strings to insert, and if you changed your value to integers, your code would work fine. The issue is how you are allocating/assigning the pointers when reading from a file then inserting it. All your pointers now point to the new value you are inserting and that's why everything changes to the new value. Same thing happened to me. I think I used a string copy function to get around this.

Interesting conjecture! This could very well be it.

But, I'm not sure how strcpy would fix things, as I already initialize a new "phrase" variable for each time I'm processing a command.

The following is my (not-so-pseudo) pseudocode for processCommand. Any ideas?

```c
// passed in a single command char, and a queue for the phrase to be processed (from main interpreter function)
```
//each item in the queue is a single character in the phrase, so we need to turn that into a char *

object

void processCommand(bst *theBST, char *command, queue *phraseQ){
  totalSize = size of phraseQ + 1

  if (totalSize == 1 and the command is not 's' or 'r') return; //empty string

  char phrase[totalSize]; //this will be what is passed into insertBST
  set phrase[totalSize-1] to null terminator;

  for (int i=0; i<totalSize-1; i++)
    phrase[i] = dequeue(phraseQ);

  switch( command){
    case 'i':
      insertBST(theBST, foo);
      break;

      ...other stuff

Subject: Re: what in the world is going on with my code
Posted by jherumin on Mon, 06 Mar 2017 01:46:29 GMT
View Forum Message <> Reply to Message

command is passed as char *, you have to switch on char or int

take away the * and make sure you're passing a single character.

Subject: Re: what in the world is going on with my code
Posted by ccatwater on Mon, 06 Mar 2017 02:55:54 GMT
View Forum Message <> Reply to Message

My issue was it happened when I was reading it into a file. So to correct it, as soon as i got the
lower-cased, correctly formatted string I said IF (its at least 1 character long) declare a new char*,
allocate space to it. Then run the string copy into the new string, then insert/delete/find the
frequency of the new string.

Subject: Re: what in the world is going on with my code
Posted by jherumin on Mon, 06 Mar 2017 03:26:11 GMT
View Forum Message <> Reply to Message
why not do

char command = string[0]

Subject: Re: what in the world is going on with my code
Posted by SSinischo on Mon, 06 Mar 2017 03:49:59 GMT
View Forum Message <> Reply to Message

jherumin wrote on Sun, 05 March 2017 19:46: command is passed as char *, you have to switch on char or int

take away the * and make sure you're passing a single character.

I don't think that's the issue. The function is being passed a pointer to my command character, which we determined just before call in my main function as it was scanning the file. Either way, my switch statement works fine and the commands are not the issue here.

ccatwater wrote on Sun, 05 March 2017 20:55: My issue was it happened when I was reading it into a file. So to correct it, as soon as i got the lower-cased, correctly formatted string I said IF (its at least 1 character long) declare a new char*, allocate space to it. Then run the string copy into the new string, then insert/delete/find the frequency of the new string.

Isn't that what I already do with char phrase[totalSize]; in the pseudocode I posted? I thought declaring a char array would allocate enough space for it. If not, what's the appropriate way to malloc an array with a variable size?