CSE 12 FALL 2015 - HW9

DUE DATE: December 6th, 11:59pm. 100 points.

PLEASE READ THIS DOC IN ITS ENTIRETY BEFORE BEGINNING

What to do:

You will be implementing a **Binary Search Tree in C**. You have been provided with started code in **BinaryTree.c.** Input/output for this assignment will be handled through the command line. The values inserted into your BST will be strings of max length 100. Your BST must implement the following functionalities:

- insert
- find
- print node values in order

To do this you will implement the following functions:

- int compare()
- void insert()
- void find()
- void traverse()

This assignment will put your knowledge of C programming to the test. **Make sure you start early!**

Function Descriptions:

int compare():

Since the values you will be inserting into your BST will be strings you cannot do a simple comparison using (<, > ==) as you could using ints. You will implement your own string comparison function that will mimic the functionality of strcmp(). compare takes two char pointers a and b

- if both strings are equal return 0
- if the first character that does not match has a lower ascii value in a than b you will return an int less than 0

- if the first character that does not match has a greater ascii value in a than b you will return an int greater than 0

DO NOT use strcmp() in the implementation of compare!

void insert():

Insert will integrate a node with an input value into the appropriate place in the tree.

- Compare must be used to compare strings (not strcmp).
- The smaller string will be the left child and the larger the right.
- If the user attempts to insert a value that is already in the tree, do nothing
- **You will need to use malloc** to allocate memory for each new node that is inserted.
- Remember malloc takes in a parameter representing the size needed in bytes. Just passing it the number of elements will not be correct.

void find():

Find will search the tree for a node containing the string the user inputs as its value.

- If the node is not found the program will output to the command line:
  "NOT found: " followed by the string the user tried to find
- If the node is found the program will output to the command line:
  "Found: " followed by the value of the found node (use printNode)

See sample output for examples

void traverse():

Traverse will output an inorder traversal of the values in the tree nodes. Use printNode to print the values of the nodes. If the tree is empty nothing will be printed.

See sample output for examples.

Development Guide:

Unlike previous assignments **you will NOT be using Eclipse to develop this assignment**. You will instead be using the text editor of your choice (GVIM, EMACS) and compiling your code through the command line (see How to Run Your Code).

THINK BEFORE YOU CODE! C's error reporting is not nearly as verbose as Java's. Often times all you'll get is a segfault. This makes debugging C code very tricky.

USE VALGRIND AND GDB! In 15L you were all taught how to use these tools. They are essential to developing and debugging in C. If you ask tutors for help they will ask if you've already tried to use these tools.

USE PRINT STATEMENTS! If you can't figure out what's going on in your code, print it out. It will make your life much easier. If you can't figure out where a segfault is happening, put print statements in the order that methods should be called, and if one doesn't show up, you now know where your error is happening.

You may choose to implement the functionalities of the binary tree in any order but the following is **recommended**:

1. compare():

You will need a strong understanding of how strings are represented in C to complete this part of the assignment. Consult your notes/online resources. You cannot use strcmp() in your implementation, however you can check your output against strcmp().

2. insert():

You should be familiar with how nodes are inserted into a BST. The tricky thing about this portion is allocating memory for new nodes using malloc. Make sure you understand how malloc works before you attempt to implement insert!

3. find():

Again something you all should be familiar with. No tricks here.

4. traverse():

This is a simple in order traversal of your BST. This can be accomplished iteratively or recursively. You are free to use helper methods here.

Your output must match the Sample Output!

Important Information:

- Do not edit any code marked "DO NOT EDIT"

- You may edit main for the purposes of testing but when you turn in your code it must match the starter code.
- You may use helper methods if you see fit. BE AWARE. In C methods must be declared before they are used.

How to Run Your Code:

1. BE ON IENG6! There is wide variety of C compilers that produce different results. Develop on ieng6 either through SSH or in the labs.
2. Navigate to the directory containing BinaryTree.c
3. type **`gcc BinaryTree.c`** into the command line. This is will produce a file called a.out IF YOU HAVE NO COMPILE ERRORS. This is your executable.
4. type **`./a.out`** into the command line to run your executable.

Alternatively, you can create files that simulate typing. For example you can have a file named “input” that contains the following

```
i
Richard
i
Andrei
i
Marina
p
```

What you can do is type **`./a.out < input`** and the program will run as though you typed that into the terminal. Also known as “piping to stdin”.

How to Submit Your Code:

Submit your completed BinaryTree.c to Vocareum.
REMEMBER: your code must compile and run on ieng6!

GRADING:

Style:
Your style will be evaluated on:

- meaningful variable names
- consistent indentation. No mixing tabs and spaces! If spaces, a consistent amount of spaces each time
- comments should be representative of your code's function. Commenting each line is not necessary, however only header comments will not suffice.

Compilation:
Your code must compile without warnings **and** errors.

Runtime:
Your program should not segfault. If your code segfaults during any of our test cases you will receive a 0 for that test case.

Consistent Output:
The output of your program must match the Sample Output provided. If you are unsure of how your program should behave given an input, consult either a tutor or post on Piazza.

Partial Credit:
You are eligible to receive partial credit. For example, if you are only able to implement compare(), **specify that in a comment at the top of your file**. You will receive some credit if it is indeed completed. *Do not expect many points for an incomplete program.*

Sample Output:

```
[cs12f6@ieng6-201]:HW:254$ ./a.out
Binary Search Tree!
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: i
Enter a string to insert (max length 100): a
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: i
Enter a string to insert (max length 100): b
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: i
Enter a string to insert (max length 100): c
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: p
a
b
```

```
c
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: q
[cs12f6@ieng6-201]:HW:255$

[cs12f6@ieng6-201]:HW:255$ ./a.out
Binary Search Tree!
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: i
Enter a string to insert (max length 100): Andrei
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: i
Enter a string to insert (max length 100): Richard
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: i
Enter a string to insert (max length 100): Marina
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: l
Enter a string to look up: Andrei
Found: Andrei
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: l
Enter a string to look up: andrei
NOT found: andrei
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: l
Enter a string to look up: Marina
Found: Marina
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: p
Andrei
Marina
Richard
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: q
[cs12f6@ieng6-201]:HW:255$

[cs12f6@ieng6-201]:HW:255$ ./a.out
Binary Search Tree!
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: l
Enter a string to look up: hello
NOT found: hello
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: p
Select an Operation: (i)nsert, (l)ookup, (p)rint in order, (q)uit: q
[cs12f6@ieng6-201]:HW:255$
```