COP3502C Computer Science 1
Dr. Andrew Steinberg
Spring 2022

Programming Assignment 4

Binary Tree Math

Max Points 100

Due: 4/1/2022 at 11:59pm

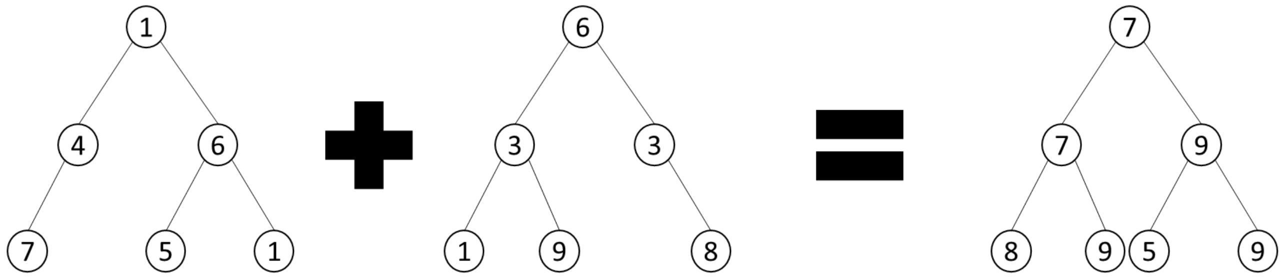
Background Story of this Assignment

You have learned about the binary tree data structure (woohoo!). Trees are an important data structure to understand as lots of companies like to ask problem solving question related to binary tree and binary search tree data structure.

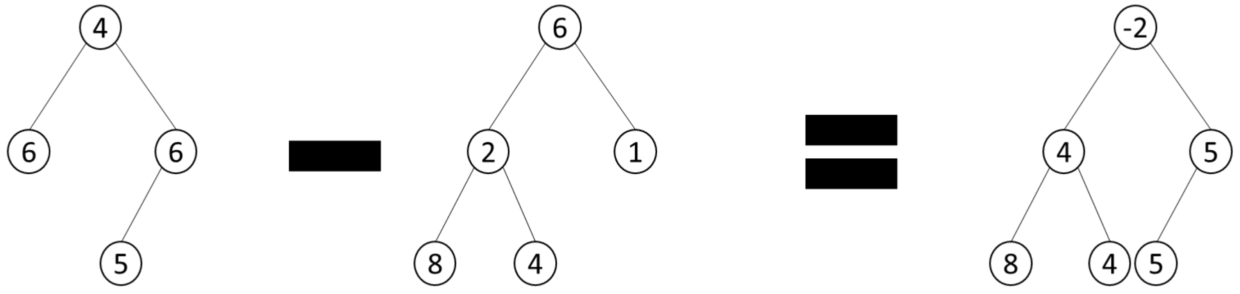
Have fun and start early (seriously start early)! Make sure to see the TAs and ULAs for help EARLY! DO NOT PROCASTINATE!

Assignment Details

In this assignment, you are going to perform some basic mathematic operations (addition, and subtraction) using binary trees (NOT BINARY SEARCH TREES!). Lets say I want to perform the addition operation with these two binary trees.



This results in a new binary tree where each node has a new value based on it's placement in the tree. Otherwise if one node is in a different location with respect to level and child, then the node is basically not modified and just placed into the new structured binary tree. Now lets say you wanted to perform subtraction between two different structured binary trees.



COP3502C Computer Science 1
Dr. Andrew Steinberg
Spring 2022

The Provided Skeleton File

For this assignment you were provided with a skeleton (program4_Skeleton.c) file that contains the following content. This section discusses the content of the provided skeleton file that you may not be familiar with.

Lines 25-38 sets up the binary trees.

Line 42 performs the addition of binary trees.

Lines 46 and 48 displays the results from the addition function.

Lines 51 – 71 repeats the same action as the previous lines except for the subtraction function.

The Function Prototypes

You were provided with a skeleton C file that has the main function and function prototypes. This section will discuss the lines of code provided for you in the main function to assist you with understanding how the code will execute. **DO NOT CHANGE THE PROTOTYPES!** Points will be deducted.

```
node_t * createNode(int val);
```

The `createNode` function will allocate a new node with the respective value to the Binary Tree. You do not need to change anything about this function.

```
node_t* insertLeft(node_t* root, int value);
```

The `insertLeft` function inserts a node manually as the left child to some respective node in the Binary Tree. You do not need to change anything about this function.

```
node_t* insertRight(node_t* root, int value);
```

The `insertRight` function inserts a node manually as the right child to some respective node in the Binary Tree. You do not need to change anything about this function.

```
void postorder(node_t * root);
```

The `postorder` display the nodes in postorder form of the Binary Tree. You do not need to change anything about this function. You do not need to change anything about this function.

```
void preorder(node_t * root);
```

The `preorder` display the nodes in preorder form of the Binary Tree. You do not need to change anything about this function. You do not need to change anything about this function.

```
node_t * addTree(node_t * t1, node_t * t2);
```

The `addTree` function takes two binary trees and performs an addition operation. The function takes two arguments which both represents roots of the respective binary trees (acting like operands). The function should return a reference to `t1` as that tree should be modified in data and structure.

```
node_t * subtractTree(node_t * t1, node_t * t2);
```

The `subtractTree` function takes two binary trees and performs an subtraction operation. The function takes two arguments which both represents roots of the respective binary trees (acting like operands). The function should return a reference to `t1` as that tree should be modified in data and structure.

Requirements

Your program must follow these requirements.

- The output must match exactly (this includes case sensitivity, white space, and even new lines). Any differences in the output will cause the grader script to say the output is not correct. Test with the script provided in order to receive potential full credit. Points will be deducted!
- Do not change ANY content of the skeleton that was provided for you. Your code will be tested through a script that relies on this main function. Any changes to this will result in your program not working fully which will lead to point deductions that will not be fixed!
- Name your C file `program4_lastname_firstname.c` where `lastname` and `firstname` is your last and first name respectively. Please make sure it matches the spelling exactly how it is registered in Webcourses. Points will be deducted if the file is not named correctly.
- You do not need to worry about dynamic allocation/memory leaks for this assignment only.
- Do not change the typedef struct that was provided for you.
- **Bonus Points Opportunity:** If you implement both functions (add and subtract) using recursion, you can receive an additional 10 points to the assignment grade. If one uses recursion and the other doesn't, then no bonus points. No partial credit will be given if your recursion implementation does not work!

Tips in Being Successful

Here are some tips and tricks that will help you with this assignment and make the experience enjoyable.

- **Draw! When coding with ADTs it is recommended you take a pencil and paper and draw your ADT after each line of code to see what you are doing in memory. This tedious and very smart method will go a super long way in coding the solution. I cannot emphasize this enough!!! It will save you from segmentation faults and losing information! It sounds tedious, but it will go a long way!!!**
- Do not try to write out all the code and build it at the end to find syntax errors. For each new line of code written (my rule of thumb is 2-3 lines), build it to see if it compiles successfully. **It will go a long way!**
- After any successful build, run the code to see what happens and what current state you are at with the program writing so you know what to do next! If the program performs what you expected, you can then move onto the next step of the code writing. If you try to write everything at once and build it successfully to find out it doesn't work properly, you will get frustrated trying find out the logical error in your code! **<u>Remember, logical errors are the hardest to fix and identify in a program!</u>**
- Start the assignment early! Do not wait last minute (the day of) to begin the assignment.

- Ask questions! It's ok to ask questions. If there are any clarifications needed, please ask TAs/ULAs and the Instructor! We are here to help!!! You can also utilize the discussion board on Webcourses to share a general question about the program as long as it doesn't violate the academic dishonesty policy.