COSC2436 Homework 3: Binary Search Trees

1. Introduction

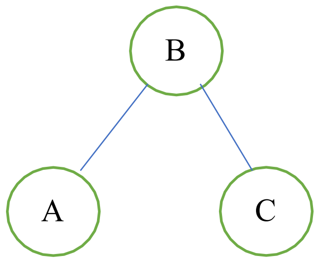
You are asked to create a simple BST in C++ to support some basic operations. In particular, you will need to support four(4) operations on the tree: Insert, Mirror, PrintLR, and PrintLevel. The program will read input from a file and perform the requested operations on the tree

2. Input files

- This file will not be empty and will contain only valid commands in the correct format.
- Be sure to remove the end-line characters (\n and \r) before processing the input strings.
- Each line in the file includes a command
- Insert(**n**) will add the item **n** to the BST
- If **n** is a string, then it should be added in alphabetical order (A goes to the left and Z to the right)
- If **n** is int, then it should be added in numerical order
- Mirror will mirror the existing BST
- PrintLR prints the BST from the most left element to the right-most element.
- PrintLevel(n) prints all the nodes in the nth level from the base/ top of the tree (base is level 0). If the level doesn't exist, simply print: "Does Not Exist!"

3. Output files

Each print function should produce a lint of output with a space between each node(" "). For instance: if the tree is



PrintLR
output: A B C
PrintLevel(0)
output: B
printLevel(1)
output: A C

Note: Initially, you will start with a BST with the smaller element to the node's left and the greater element to the right. When the "Mirror" is called, you must mirror the whole tree (hint: the left and right of each node will have to swap). Also, you will have to change the insertion function after a "Mirror" to add the smaller element to the right and greater to the left. (Hint: remember the state of the BST, normal or Mirror) If the mirror is called again, the BST and insertions should return to the "normal" (original) form.

4. Reminder

- Turn in your lab assignment to our Linux server, follow the link here for more instructions.

- Make sure to only have **one (1)** .cpp file with the main() function in your working directory, otherwise your program will fail the grading script.
    - Create a folder under your root directory, name the folder *hw3* (case sensitive), copy all your .cpp and .h files to the folder (ArgumentManager.h is also needed)
    - Only include the necessary files (.cpp and .h files) in your working directory in your final submission
    - To test your program, copy the input files into the server and run your program. After verifying that they pass, delete the .txt files.

Please reach out to myself or the TAs for any clarifications or typos.

COSC2436 hw4: BST

1. Introduction

In this homework, you are going to manipulate a BST. There will be 4 functions that you have to design. Insert, Mirror, PrintLR, and PrintLevel. Read number 5 for more information.

2. Input files

- This file cannot be empty
- You can safely consider that all the inputs are in the correct format.
- While reading the input, \n and \r should be removed before processing the string.
- Each line s going to include a command
- Insert(n) will add n to the BST
- If n is a string, then in alphabetical order (A goes to the left and Z to the right)
- If n is int, then the numerical order
- Mirror will mirror the existing BST
- PrintLR prints the BST from the most left element to the far right element.
- PrintLevel(n) prints all the nodes in the nth level from the base/ top of the tree (base is level 0). If the level doesn't exist, simply print: "Does Not Exist!"

3. Output:

Each print function should produce a lint of output with a space between each node(" "). For instance:

PrintLR
output: A B C
PrintLevel(0)
output: B
printLevel(1)
output: A C

4. Operations

Initially, you will start with a bst with the smaller number to the node's left and the greater number to the right.
When the "Mirror" is called, you must mirror the whole tree (hint: the left and right of each node will have to swap). Also, you will have to change the insertion function after a "Mirror" to add the smaller to the right and greater to the left.
It is obvious that if the mirror is called again, the bst and insertion go to the original form.

5. Requirements

Homework is individual. Your homework will be automatically screened for code plagiarism against code from other students and code from external sources. Code that is copied from another student (for instance, renaming variables, changing for and while loops, changing indentation, etc. will be treated as copy) will be detected and result in a" 0" in this homework. The limit is 50% similarity.

6. Turn in your homework

Homework 4 needs to be turned in to our Linux server; follow the link here
https://rizk.netlify.app/courses/cosc2430/2_resources/

Make sure to create a folder under your root directory, name it "hw4" (case sensitive), and copy all your .cpp and .h file to this folder, "ArgumentManager.h" need to be included as well.
PS: This document may have typos; if you think something is illogical, please email TAs for confirmation.