COSC2430 HW7

1. Introduction

You will create a C++ program that can build a AVL tree. A list of integers will be given. You need to use the integers to build a AVL tree and traverse or retrieve the data required by the commands. Step by step build your AVL tree, refer to here.

https://www.cs.usfca.edu/~galles/visualization/AVLtree.html

2. Input and Output

The inputs are regular text file, where each line is terminated with an '\n' character.

1. There are two files will be given.
    a) A data file which contains the data which will be used for the AVL tree creating. Definition: A **binary search tree**, T, is either empty or the following is true:
    i. T has a special node called the root node.
    ii. T has two sets of nodes, LT and RT, called the left subtree and right subtree of T, respectively.
    iii. The key in the root node is larger than every key in the left subtree and smaller than every key in the right subtree.
    iv. LT and RT are binary search trees.

Definition: An **AVL tree** (or height-balanced tree) is a binary search tree such that

i. The heights of the left and right subtrees of the root differ by at most 1.

ii. The left and right subtrees of the root are AVL trees.

b) A command file contains the commands, which including “Inorder Traversal”, “Preorder Traversal”, “Postorder Traversal”, “Level n” (commands are case sensitive). The n in “Level n” is a number, which represents the level of the tree, the level is start from 1, which is the root of the tree.

2. You need to check whether a line in certain file is empty.

All records should be processed sequentially from begin to end. So, the tree should be built according to the sequence of the input values.

3. Program and input specification

The main C++ program will become the executable to be tested by the TAs. The result file should be written to another text file (output file), provided with the command line. Notice that the input and output files are specified in the command line, not inside the C++ code. Notice also the quotes in the program call, to avoid Unix/Windows get confused.

Assumptions:

- For the data file, all data is integers, no characters other than digits will be

given. No empty file will be given. All integers will be separated with space characters. All integers are positive, and without sign and leading zero, but same numbers may appear more than once. The data may appear in more than one line. Also, you need to eliminate empty line. There may have empty lines between data lines.

- When creating the tree, the tree should be created with the sequence of the input data. When a value appeared before and already added as a node of the tree, you should ignore the value and proceed. When a value is a new value, you need to add it to the tree according to the principle given above. The total node number of the AVL will not exceed 1000.
- For the command file, you need to process commands one by one. One line only has one command. There will be no error command, but “Level n” may request an invalid level. There may have empty lines between data lines.
- For the output file, when output for one command, the data should be in one line without “\n”. If output for a new command, a new line should be started. If nothing required can be found, output “empty”. This only applies to the “Level n” commands.

The general call to the executable is as follows:

bitree “value=input71.txt;command=command71.txt;output=output71.txt”

Call example with another command line type.

bitree value=input71.txt command=command71.txt output=output71.txt

both type may be used simultaneously.

Example 1 of input and output

Input71.txt

2 1 3

Command71.txt

Inorder Traversal

Preorder Traversal

Postorder Traversal

command line:

bitree value=input71.txt command=command71.txt output=output71.txt

output71.txt

1 2 3

2 1 3

1 3 2

Example 2 of input and output

Input72.txt

3 10 15 23 65 85 235 457 51 9 2

235 457 51 9 2 1

Command72.txt

Preorder Traversal

Postorder Traversal

Level 3

command line:

bitree value=input72.txt command=command72.txt output=output72.txt

output72.txt

23 3 2 1 10 9 15 85 65 51 235 457

1 2 9 15 10 3 51 65 457 235 85 23

2 10 65 235

Example 3 of input and output

Input73.txt

2 6 8 45 21 63 85 55 14 16 9 3 4 7 2 55 11 13 654 214 9

command73.txt

Level 20

Level 1

Level 2

Level 3

Level 4

Level 5

command line:

bitree value=input73.txt command=command73.txt output=output73.txt

output73.txt

empty

21

8 63

3 14 45 214

2 6 11 16 55 85 654

4 7 9 13

4. Requirements

• Homework is individual. Your homework will be automatically screened for code plagiarism against code from the other students and code from external sources. If you copy/download source code from the Internet or a book, it is better for you to acknowledge it in your comments, instead of the TAs detecting it. Code that is detected to be copied from another student (for instance,

renaming variables, changing for and while loops, changing indentation, etc) will result in ”Fail” in the course and being reported to UH upper administration.

• timeout is set to 5s.

5. Turn in your homework

Homework 7 need to be turned in to our Linux server, follow the link here http://www2.cs.uh.edu/~rizk/homework.html.

Make sure to create a folder under your root directory, name it hw3 (name need to be lower case), only copy your code to this folder, no testcase or other files needed. If you use ArgumentManager.h, don’t forget to turn in it too.

ps. This document may has typos, if you think something illogical, please email TAs or Teacher for confirmation.