CMSC 3613

Programming Assignment: AVL Tree

Due Date: Check the D2L calendar for the due date.

Assignment:

The task of this project is to implement the rotate_right(), left_balance(), and avl_delete() functions in the provided program framework. Those three functions are marked as "TODO" in the comments inside the file avl_tree.h. Please follow the example program of rotate_left(), right_balance(), and avl_insert() functions, and make necessary modifications based on the code in search_tree.h. The idea is operations of AVL tree shared the same framework as BST's, so you just need to add additional code based on the functions in search_tree.h.

Please pay special attention to the balance factor for each node and the bool value longer and shorter.

The format of the input file (e.g. input1.dat) will be similar as:

```
insert
80
insert
90
insert
100
insert
110
insert
95
insert
120
insert
92
insert
97
insert
70
insert
93
delete
100
delete
80
delete
120
delete
97
```

{insert and delete will be followed by an integer in the next line. Leading or tailing spaces are not handled, if you want to make it perfect, you can follow the utilities in p01.}

The format of the output will be similar as:

```
++++++++++++++++++++
92:  90  95
90:  70  -
95:  93  110
++++++++++++++++++++
```

{This is a pre-order traversal. In each line, the format is "root_value: left_value right_value", in which the value of a null child node is replaced by a dash. Please note: the output here may not be correct, just for illustration of the format.}

Hints for avl_delete(0:

1. Deleting a node on a binary search tree is recursively solved, so it's similar for search_and_delete() in search_tree.h. The program structures will be very similar.
2. Be careful about the balance factor for each node and the bool value shorter. You can find detailed discussion in the textbook about how to remove a node from an AVL tree, please reference the contents on page 484 - 486.

Requirements:

1. Your program should follow the instructions described above in the "Assignment" section.
2. Use C++ language for implementation. The output format has been readily defined, please don't change that part.
3. The files in command line argument should be similar as follows:

   p04 input1.dat

   p04 is the name of your executable (p04 should be a fixed name!), and input1.dat is the name of the input file (the name can be any).

Evaluation:

This project will be evaluated according to the correctness of the various tree methods specified above, and secondarily according to the quality of your code. The rubric is as follows:

Categories	Weights
`rotate_right`	10%
`left_balance`	20%
`avl_delete`	30%
`code structures`	15%
`model translation`	10%
`correctness`	15%
`report`	10%

Submission:

1. Please provide a readme file, to help with the compilation and execution of your code. For example, information about your operating system and detailed command to compile your code should be included.

2. You need to submit a report by the due date on D2L. The report should include the following items:
   1) Your name and UCO email address
   2) The project number, i.e., p04
   3) A brief discussion of your implementation: just like an explanation of your idea in an interview.
   4) A screenshot of a test run.

3. Your source code also needs to submitted.

4. All the files need to be zipped as **p04_group*.zip**, where * means your group number, e.g., the submission from group 7 should be named as p04_group7.zip.

Notes:

1. To be considered on time, the program must be turned in by the due date.
2. Programs must reflect your knowledge and work, not others. You may ask others questions about algorithms, methods and programming style, but when you start writing code, you must work only with your group member(s).
3. No point, zero (0), will be given to any program containing compilation errors.