CIS*2520 Data Structures

Fall 2018

Assignment 4 Guidelines

Assignment 4 is due on Monday morning, Nov 26, 2018.

The assignment should be submitted as a tar file containing the source code as well as a readme file. There should also be a Makefile to compile the program. The compiled program should be named 'avltree'. Any warnings will result in a mark deduction appropriate to the severity of the warnings. In *readme.txt*, list everything you want to tell the TA who marks the assignment. Please remember to include your name, and student ID in all files.

1. When your program is executed, it displays a menu and a prompt when ready for keyboard input. The prompt should be string: "avl/> ". Make sure to include a blank space right after the prompt. For example:

   1. Initialization
   2. Find
   3. Insert
   4. Remove
   5. Check Height and Size
   6. Find All (above a given frequency)
   7. Exit

   Enter a code (1 – 7) and hit Return
   avl/>

2. When choice *Initialization* is selected, the program should read in the keys and create the tree. If other choices are selected before the initialization, an error message should be displayed.

3. When the user selects the second choice, the program should prompt the user for the key using the prompt 'find key: '. After the key has been found, the program should display the result in such a way: 'key: flr795, frequency: 150'. For example:

   1. Initialization
   2. Find
   3. Insert
   4. Remove
   5. Check Height and Size
   6. Find All (above a given frequency)
   7. Exit

avl/> *2*
find key: *flr795*
key: flr795, frequency: 150

or

1. Initialization
2. Find
3. Insert
4. Remove
5. Check Height and Size
6. Find All (above a given frequency)
7. Exit

avl/> *2*
find key: *monday*
no_such_key

4. Choice 3 prompts for a key string to be inserted in your AVL tree. The program should display an 'insert key: ' prompt requesting the key to be entered. After the insertion, the key and new frequency are displayed. For example

   1. Initialization
   2. Find
   3. Insert
   4. Remove
   5. Check Height and Size
   6. Find All (above a given frequency)
   7. Exit

   avl/> *3*
   insert key: *flr795*
   key: flr795, frequency: 150

5. Choice 4 prompts for the key to be removed from your AVL tree. The program should display the 'remove key: ' prompt and expect the user to enter a key. After the deletion, the key and new frequency are displayed. If the key is not found, 'no_such_key' should be printed before the program returns to the menu. For example:

   1. Initialization
   2. Find
   3. Insert
   4. Remove
   5. Check Height and Size
   6. Find All (above a given frequency)
   7. Exit

   avl/> *4*
   remove key: *flr795*
   key: flr795, frequency: 149

or

1. Initialization
2. Find
3. Insert
4. Remove
5. Check Height and Size
6. Find All (above a given frequency)
7. Exit

avl/> *4*
remove key: *tuesday*
no_such_key

6. Choice 5 prints the height and size of your tree, and total count. The output could be something like: 'height: 14, size: 22, total count: 12000'.

7. Choice 6 displays all key with a frequency above a given number. Your program should prompt for the frequency and output the keys in the same format as choice 2. For example:

   1. Initialization
   2. Find
   3. Insert
   4. Remove
   5. Check Height and Size
   6. Find All (above a given frequency)
   7. Exit

   avl/> *6*
   frequency: *20*
   key: flr794, frequency: 254
   key: flr795, frequency: 150
   key: flt123, frequency: 445
   key: flt156, frequency: 265
   ...

   You can traverse your tree in any order.

8. Choice 7 terminates your program. The changed tree is NOT required to be written back to a file.