CIS*2520 Data Structures

Fall 2014

Assignment 4 Guidelines

Assignment 4 is due on Monday morning, Nov 24, 2014.

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 tar/gz file should contain: *readme.txt*, *avltree.c*, and *Makefile*. The compiled program MUST be named 'avltree'. Failure to name your binary program 'avltree' will result in a mark deduction. Any warnings will result in a mark deduction appropriate to the severity of the warnings. There will be some marks allocated for style and documentation, but the majority will come from the correctness of the programs. 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. Make your program display a welcome banner when started. This welcome banner MUST contain your full name and student ID. After that, your program displays a menu and a prompt when it is ready for new input. The prompt must be string: "avl/> ". Make sure to include a blank space right after the prompt.

2. Choice *Initialization* should prompt for the filename on a separate line. The program should display string 'filename: ' (without quote) and expect a valid file name to be entered by the user. 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/> *1*
   filename: *A4_data_file.dat*

   Note: In the examples, user input is italic.

3. The second choice has a similar behavior as the first one. When the user selects it, 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 on a separate line. The output should be: 'height: 14, size: 22' where 14 is the actual tree height and 22 the actual size of the tree.

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.