Lab Program 3 - Updating AVL Tree to support input from STDIN & a bit more.

Michael McAlpin
Instructor - COP3502 - CS-1
Spring 2017
EECS-UCF
michael.mcalpin@ucf.edu

July 21, 2017

Abstract

Given the code for managing AVL trees is available in the program named *AVLtree.c* and the program builds its tree using programmed inputs in the *main* function the objective of this lab problem is to provide AVL tree support for a different input data type - *floating point numbers* rather than the integer support in the existing code. The updated *AVLtree.c* should be named *AVLsort.c*. The program should be able to read the unsorted data from **STDIN** and output the sorted data to **STDOUT**.

1 Objectives

1.1 Inputs

There is only one additional command parameter, the sort order (ascending or descending - *a* or *d*, respectively), passed via the command line. The data will be delivered through *STDIN*.

1.1.1 Command Line arguments

The code will be invoked as follows:

- `AVLsort a`
- Note that the command given above will read the data in from **STDIN**. Also note that the End-Of-File (EOF) key sequence will vary depending on the operating system. *This is a moot point in the event that redirection is used to input data from a simple text file.*
- The `a` *parameter* indicates the data will be sorted in ascending order. Likewise, the `d` parameter indicates the data will be sorted in descending order.

- Reading the data in from a file would be as follows:

```
AVLsort a < someFilename
```

Assuming the file redirected to **STDIN** was named SixUnsorted, the command would be as shown below:

```
//SixUnsorted contents
10.4759
99.2010
32.7510
78.3219
45.9431
25.1705

NID@Eustis$AVLsort a < SixUnsorted // The command, parameter, and redirection
```

2 Process

The program can assume that only one floating point number will be on a line and will be input from **STDIN**. In the event that the input is **NOT** a floating point number, it is acceptable to discard that number.

Each number, once succesffuly converted from text to floating point, will be added to or inserted into the AVL tree.

When all the input has been consumed, the program will then output the data, sorted in the order specified in the command line. Specifically, if the command line parameter is **a** then the numbers would be output in ascending order. Likewise, if the command line parameter is **d**, the numbers would be output in descending order.

2.1 HW 3 Considerations

This assignment uses floating point numbers for the *key*. It is well worth the while to consider using this to also code a test harness using the *three or four character airport LocID* as the *key*. We discussed in lecture that casting these *characters* as *integers* might produce meaningful alphabetically sorted results. This testing cycle, that is floating point **and** characters cast as *integers*, will streamline completing HW 3.

3 Outputs

The output of the program will be the *appropriately* sorted input data.

For the data shown above the outputs are shown below.

```
10.4759
25.1705
32.7510
45.9431
78.3219
99.2010
```

4 Grading

Scoring will be based on the following rubric:

Table 1: Grading Rubric

Percentage	Description
-100	Cannot compile on *Eustis*
- 50	Cannot accept "a" or "d" as command line argument for sort order
- 30	Cannot read input from **STDIN**
- 30	Does not sort data correctly upon completion of reading input data.

5 Submission Instructions

The assignment shall be submitted via *WebCourses*.

The source file named `AVLsort.c`. [1]

[1] With comments that confirm - "Your statement that the program is entirely your own work and that you have neither developed your code together with any another person, nor copied program code from any other person, nor permitted your code to be copied or otherwise used by any other person, nor have you copied, modified, or otherwise used program code that you have found in any external source, including but not limited to, online sources".