Name: Brian Jon Stout
Date: 04FEB2017
Current Module: Data Structures and Algorithms
Project Name: ticker

Project Goals:

The goal of the ticker program is to take in input from a file containing stock symbols, their price per stock, and the company name and insert it into a tree structure. Following this the program would take user input and either add new stocks, or update the existing one. At the end the program would read all the stocks on file, in the order of lowest stock to highest.

Considerations:

- The program must be able to sanitize data from a file, and recover from errors to avoid crashing
- The program must be able to sanitize input from a user, and recover from errors
- The program has to successfully grab data and insert it into a Binary Tree
- The tree should be an AVL and balance it self
- At the end of the program the stocks need to be printed out in order of their stock value
- The stock's price value cannot be stored as a float

Initial Design:

<A paragraph or more that adequately describes your program's structure and layout discussing the various modules (.c and corresponding .h files) that you implemented>////

Ticker.c contains the main logic code and handles maintaining data between functions. Tree.c contains all the functions related to implementing binary search trees. The functions in tree.c are declared in tree.h. Stocks.c contains functions related to sanitizing and validating input; it's functions are declared in stocks.h.

Data Flow:

The program starts by evaluating argc and argv. If argc is greater than 2 the user provided too many options in the command line argument. If argc is 2 then the program knows to read in the input file.

It first attempts to read the input file, if it can it passes the file pointer to the file_input function, along with a tree pointer called 'root' pointing to NULL. The file_input function uses fgets in a loop to read each line from the file, and passes that line to the input_validation function. The input validation function uses strtok to take each piece of the line, the first being the symbol. It checks to see if the symbol meets all the criteria. It also converts the ticker update number to a number which is converted into cents. Next it checks for the stock company

name, if there isn’t one it moves on since it isn't a requirement. If at any point the input_validation detects invalid input, it free’s up any memory allocated and stops the function. If it doesn’t it takes each value and sends it to the insert function.

The insert function contains the logic used to sort nodes in a BST based on the symbol name. It also contains the logic which makes it into an AVL. After a node is inserted, the height of each branch is checked to detect imbalances. If there is an imbalance, the function calls the correct amount of rotations to solve them.

After the function inserts the data, it falls back to the file_input function, which continues to grab each line till EOF. After all the data is loaded from the input file, the main function alls the user_input function.

The user input function is almost identical to the file_input function, except the look breaks on a single newline, and uses STDIN instead of a file pointer. The user_input function also passes the inputted line into the input_validation function.

After the user_input is finished, the main function creates a new tree pointer called newTree. newTree is going to contain a BST sorted by cents value as opposed to the symbol name. The main function calls the sort_tree function with the original tree. The sort_tree function has a loop that repeats the dismantle function until dismantle returns NULL. The dismantle function returns an individual node from a tree until every node has been retrieved only once. When the dismantle function returns a node, it sorts it into a new tree using a separate insert function that inserts by cents value and not Symbol. At the end the sort_tree function returns a tree pointer to the newly sorted tree.

The main function then prints out the tree in a LPR order so the smallest stock values come out first. On completion the program runs the tree_destroy functions on both trees to clean up all memory and exits.

Communication Protocol:

Not applicable.

Potential Pitfalls:

The biggest issue is deciding what the key will be for the binary search tree. The main lookups will be done with the symbol which provide a decent key because they don’t change, whereas the stock price can. Sorting a tree by a different key requires it to be rebuilt.

Test Plan:

User Test:

User add stocks manually without initial file. Attempts some bad input.
User specifies file to be used. Skips user input.
User specifies files and provide input and updates.

Different stock updates to put the values at different places, including negative values.
Skipping input altogether.
Providing file with bad input.
Providing bad user input.

Test Cases:

All the test cases provide some meaningful response for errors, and there are no memory leaks.

Conclusion:

The project's main challenge was sanitizing all the input. Pushing all the stocks in an AVL tree based on the symbol wasn't too difficult, but some issues were ran in to later when the info needed to be sorted by the stock value instead. Using a hash lookup table for the symbol which points to the node in the BST which is already sorted by the stock value. If I had time I would have definitely used two different structures, one to look up a stock quickly by it's node, and the other which keeps it pre sorted by the value.

Using a doubly link list would probably be the best since you can shift left and right fairly easy for updated values. Optimizing the BST to use hashes from the symbol would probably be a lot quicker than using strncmp since strncmp has to analyze every character of the a string with every character of a different string every time it steps into.

However implementing custom structures requires a lot more thought and experimentation, which requires a lot more time. Breaking down the tree and putting it into another one isn't the most elegant solution, but it still works decently.