CSC 520
Programming Project 2
Binary Search Trees

Due: ***Tuesday*** October 26, 2010 11:59 PM **(**Design of data structure due October 13, 2010)

In this project you will design and implement a data structure to maintain a set of stock prices. Note that several stocks might have the same price (i.e. prices are not unique). Your data structure should support operations

1. Insert: insert a new stock and its price -- given name and price
2. Delete: delete a stock -- given a pointer to the object representing this stock
3. Change: change a stock price -- given stock name and a new price
4. Find: return a pointer to the object representing this stock – given the stock name.
5. FindMax: find and output the most expensive stock, it there are more than one such stock, just output one of them
6. Range: compute and output which percentile the price of this stock lies in – given the name of the stock.
   For example, if the stock prices are 1.40, 2.60, 2.60, 3.70, 3.70, 3.70, 3.70, 4.30, 4.30, 5.0 then a stock with price 3.70 is between 4/10 and 7/10. You need to output both of these numbers.
7. HighAverage: compute and output the average of stock prices equal to or larger than a given price
8. ShowAll: output all the stocks in alphabetical order together with their prices.

Operations 1-7 should run in $O(\log n)$ time, where n is the number of stocks. Operation 8 should run in $O(n)$ time.

You probably will use binary search trees or some variation of binary search trees to store the stocks. In order to achieve a time complexity of $O(\log n)$ time, where n is the number of stocks, we must make sure that the binary search tree is height balanced. However, due to time constrain, we will make this optional. Your operations just need to run in $O(h)$ time, where h is the height of the binary tree

.

Hand in

October 13: A description of the data structures you are going to use including the fields of the nodes, and a brief description of how to implement the operations.

October 26:
1) A hard copy of a well-documented program.
2) A copy of your test data with comments indicating why a command is included in the test
3) A copy of the output from the execution of your program on your test data in 2).
4) Submit a file containing the source code of your program to the class Blackboard.