CS 260
Spring 2015
Project 6

- Write a Python3 program project6.py that operates as described below.
- Your program should accept three command line arguments. The first argument is the name of an input file, and the remaining two arguments are the names of the output files.
- Recall that we previously computed capital gains on stock transactions in Project 4. Now we consider two new accounting methods for calculating a cost basis. These two accounting methods are illustrated via an example:

Series of transactions	**Min cost method**	**Max cost method**
Buy 20 shares of IBM at $3.00 per share		
Buy 20 shares of IBM at $7.00 per share		
Buy 40 shares of MSFT at $5.00 per share		
Buy 50 shares of MSFT at $4.00 per share		
Buy 20 shares of IBM at $2.00 per share		
Buy 20 shares of IBM at $6.00 per share		
Sell 30 shares of IBM at $9.00 per share (Sales price = 30*9 = 270)	Cost basis = 20*2 + 10*3 = 70, Capital gain = 270 - 70 = 200	Cost basis = 20*7 + 10*6 = 200, Capital gain = 270 - 200 = 70
Buy 10 shares of IBM at $1.00 per share		
Buy 10 shares of IBM at $8.00 per share		
Sell 10 shares of MSFT at $15.00 per share (Sales price = 10*15 = 150)	Cost basis = 10*4 = 40, Capital gain = 150 - 40 = 110	Cost basis = 10*5 = 50, Capital gain = 150 - 50 = 100
Sell 20 shares of IBM at $10.00 per share (Sales price = 20*10 = 200)	Cost basis = 10*1 + 10*3 = 40, Capital gain = 200 - 40 = 160	Cost basis = 10*8 + 10*6 = 140, Capital gain = 200 - 140 = 60

- Your program will take as input a series of stock transactions, and it will produce as output the capital gain (or loss) for each sale, as well as the total capital gain (or loss) for the entire series of transactions.
- Your program should use two kinds of priority queues: a minimum-ordered heap and a maximum-ordered heap. Each heap node should hold a pair that consists of (number of shares, price per share).
- Your program should also use a binary search tree to represent a dictionary, where the primary keys are the stock ticker symbols. The dictionary should map each stock ticker symbol to the appropriate pair of priority queues. Note that each minimum or maximum cost is determined with respect to shares of stock from a single company, so currently-owned shares of stock from different companies must be maintained in separate priority queues.

- Example: You can run your program with this command:
  python3 project6.py Transactions1.txt MinCost1.txt MaxCost1.txt
  Here are input and output files that correspond to the previous example:

Transactions1.txt	**MinCost1.txt**	**MaxCost1.txt**
Buy 20 IBM 3.00 Buy 20 IBM 7.00 Buy 40 MSFT 5.00 Buy 50 MSFT 4.00 Buy 20 IBM 2.00 Buy 20 IBM 6.00 Sell 30 IBM 9.00 Buy 10 IBM 1.00 Buy 10 IBM 8.00 Sell 10 MSFT 15.00 Sell 20 IBM 10.00	Gain = 200.00 Gain = 110.00 Gain = 160.00 Total = 470.00	Gain = 70.00 Gain = 100.00 Gain = 60.00 Total = 230.00

- Each line of the input file (Transactions1.txt) contains a string ("Buy" or "Sell"), an integer (the number of shares), another string (the stock ticker symbol), and a floating point value (the price per share).
- Each line of the output files (MinCost1.txt, MaxCost1.txt) corresponds to a "Sell" transaction, and consists of either "Gain =" followed by a positive floating point value, "Loss =" followed by a negative floating point value, or just "Zero". The final line of each output file shows the total net gain (or loss) over all the "Sell" transactions.
- Display each floating point value with exactly two digits after the decimal point, by using "{:.2f}".format(value) to obtain the formatted string representation of the value to be displayed.

The following rules apply to all projects in this course:

- We now provide a virtual machine so you can test your program in the same environment where it will be graded. This server is "cs260.ua.edu". Regardless where you primarily develop your program, you must use this server to test your projects before submitting. Login using your Bama userid and password. You might first need to install the SSH secure shell and file transfer clients, for example at http://helpdesk.ua.edu/~recovery/ssh.htm. Also, if you wish to access the server from off-campus, you'll need to install a VPN client, which is available at https://mybama.ua.edu on the Tech tab.
- During grading, your program will be tested on the cs260.ua.edu server. Example scripts and input/output files will be provided on this server in the /projects directory.
- **You are responsible for ensuring that your program works correctly for all possible inputs. The provided input/output files are NOT guaranteed to catch every error in your program. We will test your program using other data that was not provided to you in advance. So you should read the assignment carefully, do not make any extra assumptions about the I/O formats, ask for clarification if you are in doubt, and thoroughly test your program by creating additional data beyond the files we provide.**

- We provide a common platform that all students must use to test their programs before submitting, and we will use this same platform to evaluate each submitted program. Your program will NOT be evaluated based on how it performs in any other environment, because we will not attempt to replicate different environments used by each individual student.
- In previous semesters, some students did not follow the instructions and/or did not adequately test their programs before submitting, so they lost points for errors that they could have easily detected and corrected. This semester, it is required to test the final version of your program on the server before you submit it. All the scripts and input/output files are provided on the server in the /projects directory. When you run the script, it will produce a file "checksums.txt" that you must submit along with your program. When grading your project, we will use the contents of this file to verify that you actually did test the final version of your program on the server.
- You are permitted to verbally discuss ideas with classmates or other people, but sharing any code is strictly forbidden. **Do not look at your classmates' or anybody else's code, and do not show anybody your code!** However, you are encouraged to verbally discuss the projects with your classmates, provided nobody involved is looking at any code during the discussion.
- You must write all your own code for each project. Do not take code from any other source. If you happen to find a similar program online or in a book, you are NOT permitted to copy portions of that program into your program. The purpose of the projects is to learn how to write programs for data structures and algorithms, not to practice googling for solutions.
- I**f your code is too similar to another student's code or to anybody else's code, you will receive an invitation to discuss the situation in the Dean's office.** We use the MOSS system which detects plagiarism based on programs that are too similar. The penalty for plagiarism is typically a score of 0 on the project or a grade of F in the course.
- You should strive to make each program as efficient as possible. During testing, we will allow each program to run for a reasonable amount of time, but if your program takes much longer to run than we think it should, we will eventually decide your program is too inefficient (or possibly in an infinite loop), and in this case we may terminate the execution of your program before it ends.
- Place all your program's source files and the file "checksums.txt" (generated by the script) into a single zip file, give it a name of the form LastName_FirstName_ProjectNumber.zip (example: Doe_John_1.zip), and upload this zip file using Blackboard. Make sure your zip file includes the latest version of all your source files. Your zip file should contain the file "checksums.txt" and all the source files needed to run your program, and these files should all be located at the top level of your zip file. The zip file should not contain any other subfolders. Do not submit projects via email.
- Each assignment will be due by 11:59pm on the due date specified on Blackboard. Please aim to submit your project earlier than the due date in case you encounter unforeseen circumstances at the last minute. There will be a substantial deduction for late submissions. Points will also be deducted if you do not follow all the instructions exactly, because this can make it difficult or impossible to grade your project.

- After the projects are graded, your score will be uploaded onto Blackboard, along with a grading report that summarizes how your score was calculated. Also, any additional input/output files used when grading the projects will be placed in the /projects directory on the `cs260.ua.edu` server.
- To completely understand your project's score, you will likely need to run your program on the server using the new input/output files. If you think your project grade was computed incorrectly, you must inform us **within at most one week** from when your score is posted on Blackboard. You must stop by one of the instructors' offices during office hours and show us that your program runs correctly using the script and input/output files located on the server. If a mistake was made when grading your program, we will be glad to correct it.
- However, note that the following statements (which we hear quite frequently) will NOT cause us to increase your project score:
  - "I spent many hours working on the project, so I want a higher score."
  - "My program contains over 1000 lines of code, so I want a higher score."
  - "I didn't follow all the instructions because I thought they were optional."
  - "My program shouldn't be tested on any valid inputs that I forgot to consider."
  - "I didn't understand the project description, and I didn't ask for clarification."