## Computing in Finance (Fall 2017) – Summer Assignment

The objective of this assignment is to make sure that everyone entering Computing in Finance course has a very basic familiarity with Java. You should be in a position to submit this assignment via email in the first class session. (You will be told how to name and submit your work, so do not try to submit it before the first class.)

It is very important that you take this assignment seriously. Not only will the assignment count towards your grade, but the course will assume a programming ability that starts at this level. If you are already familiar with Java, you should find these exercises easy. Otherwise, you can do these exercises as you make your way through a Java programming book. (We suggest a book for your use.)

## Running a Program from Command Line

Write a "hello world" program that runs from the command line of your computing environment. You may be using an IDE (Integrated Development Environment) to do your development. However, it is important to be able to get your program to run from the command line as well as from the IDE. (All Java books explain how to do this.)

Your "hello world" program should output: <Your name>, Financial Mathematics, Courant Institute

## Manipulating Data

We have attached two text files. The first – prices.csv – contains daily price data for SPY, an ETF (Exchange Traded Fund) that represents ownership of the Standard and Poor's 500 stock composite price index. The second file – corrections.csv – contains a list of corrections and additions meant to modify data in the first file. (More on this later.)

Please write a Java class called DataHandler that has the following methods. Note that all references to price are to the prices that appear in the Adj Close column of the data file.

**loadPriceData**
This function should take the file name, sort method specification (QuickSort or BubbleSort), and sort preference – ascending or descending, and price or date – as parameters. The file content should be loaded into memory and sorted according to the above sort preferences. After this method is called, the data should be held in the instance variables of an object of class DataHandler.

**getPrices**
This method should take the parameters fromDate and toDate. It should return an array or a list of prices for the period specified the two dates. The prices returned should include prices for the start date, the end date, and everything in between.

**computeAverage**
This method should take the parameters fromDate and toDate. It should return the average price for the period specified by the two dates. The calculation of average price should include prices for the start date, the end date, and everything in between.

**computeMax**
This method should take the parameters fromDate and toDate. It should return the maximum price for the period specified by the two dates. The calculation of maximum price should include prices for the start date, the end date, and everything in between.

**computeMovingAverage**
This method should take the parameters windowSize, fromDate, and toDate. It should return a moving average of the last n elements of price, where n = windowSize. The prices used in this calculation should include prices for the start date, the end date, and everything in between. For example, if the dates specified are from the 1st of July to the 20th of July and the window size is 10, this method should return an array or list in which the 1st value is the average of the price from the 1st of July through the 10th of July, the 2nd value is the average of the price from the 2nd of July through the 11th of July, and so on, until the last value used in the calculation is the price from the 20th of July.

**insertPrice**
This method takes a price record as a parameter. As in the price file, a single price record has a date, open price, high price, low price, close price, trading volume, and adjusted closing price. The insertPrice method is used to insert a price into the price data contained in an object of class DataHandler. If the date already exists, overwrite the record that is already there. Note that you must keep the data sorted when inserting or overwriting a new price record.

**correctPrices**
This method takes the file name of the corrections file as a parameter. The corrections file looks just like the price file. This method can call the insertPrice method to perform the actual corrections.

**main**
Write a main method for class DataHandler. In your main method do the following.

1) Create an object of class DataHandler

2) Call its loadPriceData method to load price data from the prices.csv file. Specify that the data is sorted by date using a Quick Sort and should be stored in ascending order.

3) Call your DataHandler object's correctPrices method specifying the corrections.csv file as the source of the corrections.

4) Output the following to a results.txt file. “The Prices of SPY between 08/15/2004 and 08/20/2004 are: “. Retrieve the appropriate prices for the specified dates using the getPrices method and append them to the results.txt file.

5) Append the following to the results.txt file. ““The Average Price of SPY between 08/15/2004 and 09/15/2004 is: “. Retrieve the average price for the specified dates using the computeAverage method and append it to the results.txt file.

6) Append the following to the results.txt file. “The Maximum Price of SPY between 04/15/2004 and 06/15/2004 is: “. Retrieve the maximum price for the specified dates using the computeMax method and append it to the results.txt file.

7) Append the following to the results.txt file. “The Moving Average of SPY between 08/15/2004 and 09/15/2004 for WindowSize 10 is : “. Retrieve the moving average for the specified dates using the computeMovingAverage method and append the results to the results.txt file.