COS 126 General Computer Science Fall 2017

Programming Exam 1

Instructions. This exam has two parts and an optional extra credit. You have 50 minutes. The exam is *open course materials*, which includes the course textbook, the companion booksite, the course website, your course notes, and code you wrote for the course. Accessing other information or communicating with a non-staff member (such as via email, text, Facebook, Piazza, phone, or Snapchat) is prohibited.

Setup. You may download the necessary files from the *Class Meetings* page now.

Submission. Submit your solution via the link on the *Class Meetings* page. Click the *Check All Submitted Files* button to verify your submission. You may submit multiple times.

Grading. Your program will be graded for correctness, clarity (including comments), design, and efficiency. You will receive partial credit for a program that correctly implements some of the required functionality. You will receive a substantial penalty if your program does not compile or if you do not follow the prescribed input/output specifications.

Discussing this exam. Discussing or communicating the contents of this exam before solutions have been posted is a violation of the Honor Code.

This exam. You must turn in this exam. Print your name, NetID, precept, and the room in which you are taking the exam in the space below. Also, write and sign the Honor Code pledge. You may fill in this information now.

Name:

NetID:

Exam Room:

Precept:

"I pledge my honor that I will not violate the Honor Code during this examination."

Signature

Problem 1: Plot Prices. (20 Points) Write a program `Prices.java` that reads, prints, and plots stock prices. Your program must handle all inputs that conform to the input specification. We will not test non-conforming inputs. See **Reminders and Tips** on p. 4.

Input Specification. The input from standard input consists of a positive integer n on a line by itself, followed by a sequence of n lines, each representing one day of historical stock trading information. Each line contains a date string and four prices (doubles) for that day's trading. The four prices are the *open* (the price of the first trade that day), the *high* (the highest trade price for that day), the *low* (the lowest trade price for that day), and the *close* (the price of the last trade of the day).

```
% more TSLA-10.txt
10 ← n
2013-01-02 35.00 35.45 34.71 35.36
2013-01-03 35.18 35.45 34.75 34.77
2013-01-04 34.80 34.80 33.92 34.40
2013-01-07 34.80 34.80 33.90 34.34
2013-01-08 34.50 34.50 33.11 33.68
2013-01-09 34.01 34.19 33.40 33.64
2013-01-10 33.87 33.99 33.38 33.53
2013-01-11 34.04 34.04 32.11 32.91
2013-01-14 33.08 33.38 32.85 33.26
2013-01-15 33.11 34.25 33.08 33.90
   date       open  high  low  close
```

Two data files, `TSLA.txt` (Tesla Inc. stock prices for 2013) and `TSLA-10.txt` (Tesla Inc. stock prices for the first 10 days of 2013), are available via the *Class Meetings* page.

Required Program Steps.

1. Read n, the number of days, from standard input.
2. Read, also from standard input, the date and the open, high, low, and close prices for each day into five parallel arrays.
3. Call `StockPlot.plot` to make a pretty plot.
4. Print n and the array contents to standard output.

Standard Output Specification. The output should match the input exactly. Use `%.2f` in the `StdOut.printf` format string for the doubles. Do *not* print any other output.

Calling `StockPlot.plot`. Plot the stock prices using the `StockPlot.plot` function provided to you in the `StockPlot.java` file available from the *Class Meetings* page. The function `StockPlot.plot` takes four array arguments of the same size: `open`, `high`, `low`, and `close`. The plot generated is called a "candlestick plot" with the range between open and close in red and the remainder of the range between high and low in black.

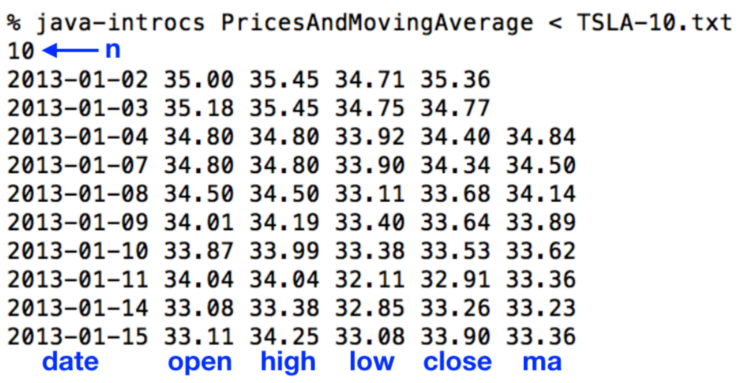
Submission. Submit `Prices.java` via the link on the *Class Meetings* page. Tip: Successfully complete Problem 1 before moving on. Problem 1 is worth the most points.

Problem 2. Simple Moving Average (10 Points) Write `MovingAverage3.java`, a program that reads stock prices, computes the 3-day simple moving average of the *close* prices, plots the prices and moving average, and prints the prices and moving average to standard out. Your program must handle all inputs that conform to the input specification. We will not test non-conforming inputs.

Computing the 3-day Simple Moving Average In finance, a moving average is a very popular stock price analysis indicator. It provides a smooth trend line, removing the distractions of higher frequency price fluctuations. In this problem, you will compute the simple moving average of the closing stock price. The simple moving average on day i is the average of the three most recent *close* prices, including the *close* price on day i. You can compute the average for day i by summing the *close* price for days i, $i-1$, and $i-2$ and dividing by 3. Do not compute the moving average for i = 0 and i = 1.

Input specification. The input is the same as Problem 1. The input will have at least three days of trading history.

Standard output specification. The output should match the input exactly except with the addition of a new column containing the 3-day simple moving average starting on the 3rd day. (The 3-day moving average has no meaning on the first two days.) Do *not* print any other output to standard output.



```
% java-introcs PricesAndMovingAverage < TSLA-10.txt
10
2013-01-02 35.00 35.45 34.71 35.36
2013-01-03 35.18 35.45 34.75 34.77
2013-01-04 34.80 34.80 33.92 34.40 34.84
2013-01-07 34.80 34.80 33.90 34.34 34.50
2013-01-08 34.50 34.50 33.11 33.68 34.14
2013-01-09 34.01 34.19 33.40 33.64 33.89
2013-01-10 33.87 33.99 33.38 33.53 33.62
2013-01-11 34.04 34.04 32.11 32.91 33.36
2013-01-14 33.08 33.38 32.85 33.26 33.23
2013-01-15 33.11 34.25 33.08 33.90 33.36
```

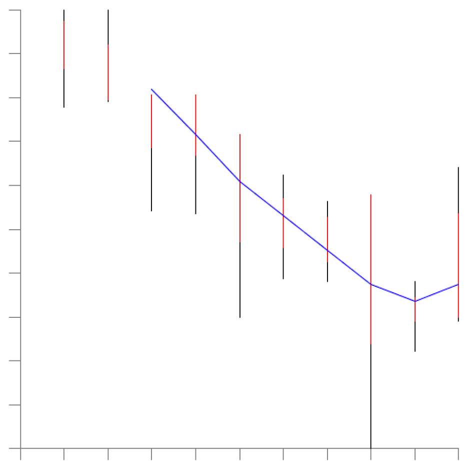
Plotting. The plotting will be performed by `StockPlot.plot`, the same function used in Problem 1. To plot the moving average, pass six arguments instead of four. These arguments are `open`, `high`, `low`, `close`, `ma` (the moving average), and `days`. The first five arguments passed must be double arrays of the same size. The last argument, `days`, an integer, expresses the moving average number of days (3 in this case). `StockPlot.plot` will ignore the values in `ma` for days less than `days` (index 0 and 1 in this case).

Submission. Submit `MovingAverage3.java` via the link on the *Class Meetings* page.

Reminders and Tips

- Be sure to compile your Java files with `javac-introcs`.
- Be sure to also compile StockPlot.java with `javac-introcs`.
- Be sure to run your program with `java-introcs`.
- Tip: Save time on this exam by copying your code from one part to the next.
- Tip: Do not move to the next part until your submitted code passes all tests.

Sample Plot. `java-introcs MovingAverage3 < TSLA-10.txt` should produce this plot:



Optional Extra Credit. (2 Points) Write a program `MovingAverageK.java` that reads stock prices, computes the K-day simple moving average of the *close* prices, plots the prices and moving average, and prints the prices and moving average to standard out. K is an integer argument on the command line. Your program must handle all inputs that conform to the input specification. We will not test non-conforming inputs. The input will have at least K days of trading history.

Output specification. The output printed should match the input exactly except with the addition of the K-day moving average starting on the Kth day. (The K-day moving average has no meaning on the first K-1 days.) Do *not* print any other output to standard output. Plot using `StockPlot.plot`.

Submission. Submit `MovingAverageK.java` via the link on the *Class Meetings* page.