SMA Crossover

What we are looking for

We request all of our candidates for Engineering roles complete a simple coding test in their own time to be submitted for review. What we hope to get from your solution:

- Correctness based on the problem description & examples
- Good code quality & development practices
- Ease of maintenance
- In-code documentation & comments where required
- Good quality tests

This test will be used to inform some of the questions we ask in the first technical interview, so being able to explain the decisions and assumptions behind your solution is critical. We understand some of these criteria are subjective so we are not looking for the perfect solution, but to understand how the candidate thinks about things like code quality and testing.

Problem Description

The task is to compute two Simple Moving Averages (SMAs) on a single time series of closing stock prices for a single stock and use these to inform a simple trading strategy.

An SMA is a rolling average of fixed window size, calculated at the end of the window. As an example, if we have these 6 numbers:

```
13.0,18.0,18.0,20.0,24.0,22.0
```

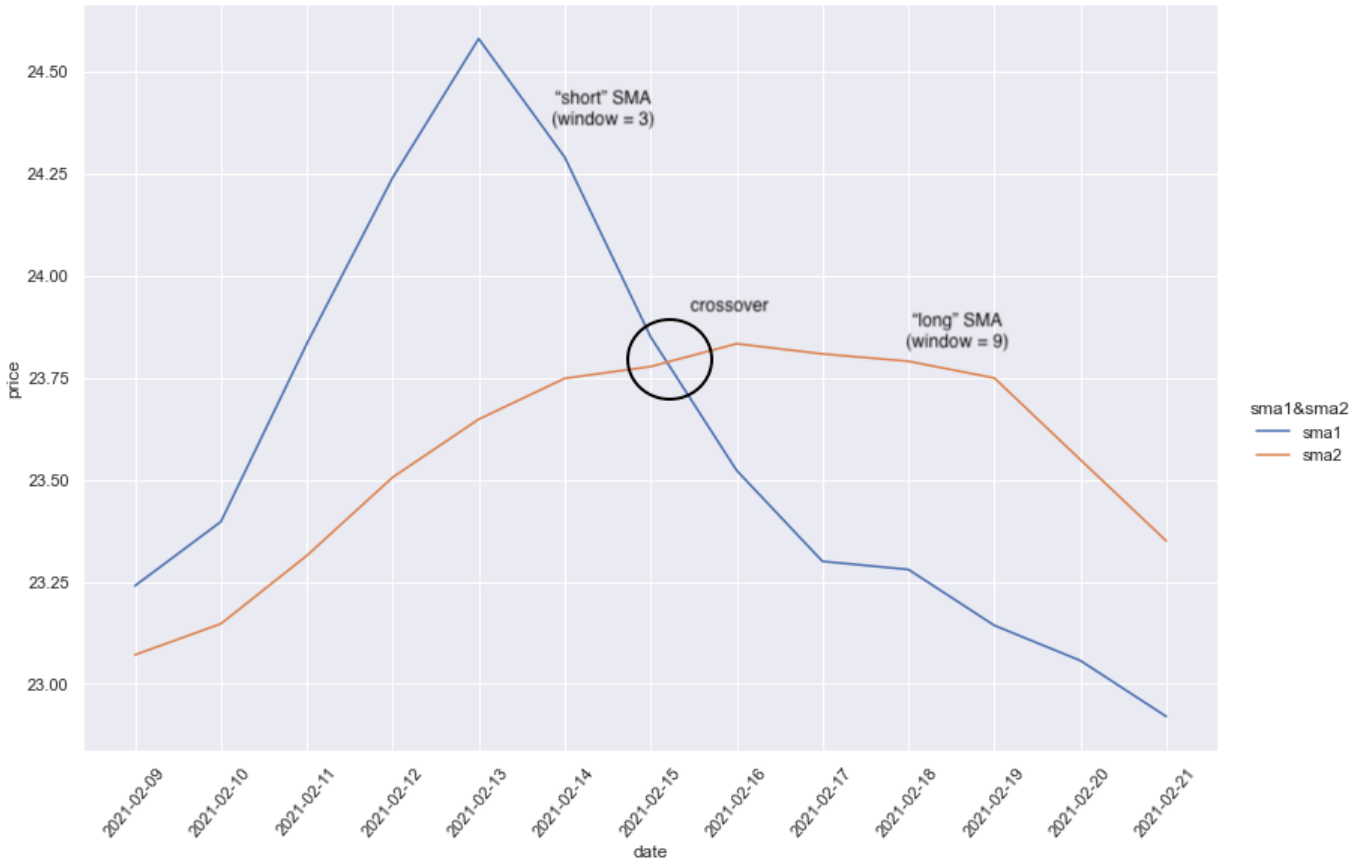
And a window 5 days then the SMA at the end of day 5 is:

```
(13.0 + 18.0 + 18.0 + 20.0 + 24.0) / 5 = 18.6
```

And the SMA at the end of day 6 is:

```
(18.0 + 18.0 + 20.0 + 24.0 + 22.0) / 5 = 20.4
```

The way the strategy works is we compare two SMAs of the same time series with different window sizes, one shorter than the other. If the "shorter" SMA crosses the "longer" SMA it is a buy signal (which we represent as `1`). When the "longer" SMA crosses the "shorter" SMA it is a sell signal (which we represent as `-1`). A `0` signifies we don't yet have enough data for a buy or sell signal.



A detailed description of this and other moving averages strategies is available at https://www.investopedia.com/articles/active-trading/052014/how-use-moving-average-buy-stocks.asp but you should only need information in these instructions to implement the code.

Example

Here is an example input of a CSV with dates and the closing price of a stock:

```
date,close
2021-02-01,22.81
2021-02-02,23.09
2021-02-03,22.91
2021-02-04,23.23
2021-02-05,22.83
2021-02-06,23.05
2021-02-07,23.02
2021-02-08,23.29
2021-02-09,23.41
2021-02-10,23.49
2021-02-11,24.60
2021-02-12,24.63
2021-02-13,24.51
2021-02-14,23.73
2021-02-15,23.31
2021-02-16,23.53
2021-02-17,23.06
2021-02-18,23.25
2021-02-19,23.12
2021-02-20,22.80
2021-02-21,22.84
```

If we compute a short SMA with a window of 3 days and a long SMA with a window of 9 days we should produce a CSV file that looks like this:

```
date,close,sma1,sma2,position
2021-02-01,22.810,,,0
2021-02-02,23.090,,,0
2021-02-03,22.910,22.937,,0
2021-02-04,23.230,23.077,,0
2021-02-05,22.830,22.990,,0
2021-02-06,23.050,23.037,,0
2021-02-07,23.020,22.967,,0
2021-02-08,23.290,23.120,,0
2021-02-09,23.410,23.240,23.071,1
2021-02-10,23.490,23.397,23.147,1
2021-02-11,24.600,23.833,23.314,1
2021-02-12,24.630,24.240,23.506,1
2021-02-13,24.510,24.580,23.648,1
2021-02-14,23.730,24.290,23.748,1
2021-02-15,23.310,23.850,23.777,1
2021-02-16,23.530,23.523,23.833,-1
2021-02-17,23.060,23.300,23.808,-1
2021-02-18,23.250,23.280,23.790,-1
2021-02-19,23.120,23.143,23.749,-1
2021-02-20,22.800,23.057,23.549,-1
2021-02-21,22.840,22.920,23.350,-1
```

Instructions

1. Implement a program that will:
    a. Read a CSV file in the example input format given above
    b. Take two window sizes in days, one for SMA1 (the "short" SMA) and one for SMA2 (the "long" SMA)
    c. Compute the two SMAs and determines which position (long or short) to take
    d. Output a CSV in the example output format given above
2. This program can be written in any of the following languages:
    a. Python (preferred)
    b. Java, Scala or Kotlin
    c. Javascript or Typescript
3. You may use any frameworks or libraries internally you are comfortable with but we ask you **implement the core SMA algorithm from scratch**. You can also use whatever build tools etc you are most comfortable with.
4. The example project should include some level of unit testing using whatever testing frameworks you are comfortable with. We do look at test coverage, however we don't expect 100% coverage. We will have a discussion on the decisions you made on what level of testing to include in your solution.
5. You should outline any assumptions you have to make if you don't feel they are clearly articulated in the description or example above.
6. The project should include three shell scripts that work as follows:
    a. `build.sh`: This script takes no arguments and should setup whatever you need to run the code. There should not be any other manual steps required to run the program once this script has been run.
    b. `test.sh`: This script takes no arguments and should run your unit tests and output to standard out (STDOUT)
    c. `run.sh $INPUT_CSV $SMA1_WINDOW $SMA2_WINDOW $OUTPUT_CSV`: This script takes 4 arguments and should run your program. NOTE that we will use this `run.sh` script to test your program with our own test data to validate it runs correctly.
7. Create a **PRIVATE** Github repo (please, **no public repos**) and use the following instructions to add the user rzt-recruitment-github@rozettatechnology.com as a collaborator to your repo: https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/inviting-collaborators-to-a-personal-repository
8. We will require access while we review your code so please don't remove collaborator permissions until we have contacted you to do so.
9. Take a screen shot of the `Settings  Manage Access` screen of your repo that includes list of collaborators invited under `Manage Access`. Send these screenshots back to the RoZetta HR contact that sent you this coding test to confirm your completion.