CS 5007 Individual Project Assignment 2 (100 Points)

Due: 11:59 p.m. on 10/08/2021

Project Objective: The goal of this project is to develop and implement an application using the functional programming paradigm.

Project Deliverables:

1. Submit your hierarchy chart (**.pdf**) and the **.py** file to Canvas.
2. Provide the clear **comment documentations** to your code.

Note:

(1) This project is to be done by each student individually. No help besides the textbook, materials, and the instructor should be taken. Copying any answers or part of answers from other sources (including your classmates) will earn you a grade of zero.
(2) All programming conventions mentioned in class should be followed.
(3) You should test your program before submitting.
(4) Your program must be developed and implemented in the PyCharm IDE, or 10% of the graded score is deducted.
(5) Assignments are accepted in their assigned Canvas drop box without penalty if they are received by 11:59PM EST on the due date, or 10% of the graded score is deducted for the late submission per day. Work submitted after one week of its original due date will not be accepted.

Task:

Data mining is the process of sorting through large amounts of data and picking out relevant information. It is usually used by business intelligence organizations and financial analysts but is increasingly being used in the sciences to extract information from the enormous data sets generated by modern experimental and observational methods.

In this project, we want to do some preliminary data mining to the prices of some company's stock. So that we can speak in specifics, let's look at Google. Google has two listed share classes that use slightly different ticker symbols:

- GOOGL shares are its Class A shares, also known as common stock, which have the typical one-share-one-vote structure.
- GOOG shares are Class C shares, meaning that these shareholders have no voting rights.

Your application will calculate the monthly average prices of Google stock from Jan 01, 2016 to Dec 31, 2020 and tell us the six best and six worst months for Google within these five years. In this application, you need to use the GOOGL shares **NOT** GOOG shares.

First you need a history of stock prices. Go to finance.yahoo.com, enter **Googl NOT** Google and Goog in the search field, select "Historical Data" and choose the time period between Jan 01, 2016 to Dec 31, 2020 with the frequency "Daily". Click on the "Download" button and save this **raw .csv** file into the folder where you will develop and implement your application.

There are seven functions in this program shown in the below table.

Function List

Function	Description
get_data_list()	Read the raw .csv file and return the below **2D list** that contains all the daily stock prices for Google from Jan 01, 2016 to Dec 31, 2020. **[['Date', 'Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume'], ['1/4/2016', '762.200012', '762.200012', '747.539978', '759.440002', '759.440002', '3369100'], ['1/5/2016', '764.099976', '769.200012', '755.650024', '761.530029', '761.530029', '2260800'],** **:** **:** **]**
get_monthly_average()	Get the **2D list** above as the input from get_data_list(), compute the monthly average prices in each year using **Date**, **Volume**, and **Adj Close** fields, and then return the below list. Here is the formula for the average price of a month, where Vi is the volume and Ci is the day's adjusted close price (Adj Close). **monthly_average_price = (V1 * C1 + V2 * C2 + … + Vn * Cn) / (V1 + V2 + … + Vn)** **[['1/2016', 736.5629660252736], ['2/2016', 729.511817814572], ['3/2016', 744.1192538757223], ['4/2016', 750.1933494928849],** **:** **:** **]**
find_six_highest_lowest_monthly_average_price()	Get the **2D list** above as the input from get_monthly_average(), find and return the six highest monthly average prices and the six lowest monthly average prices for Google's stock.
reformat_max_res()	Get the **2D list** that contains the six highest monthly average prices only as the input from find_six_highest_lowest_monthly_average_price(), reformat the date as follows, e.g., **11/2020 → Nov 2020**, and return a new list with the new date format, e.g., from **11/2020: 1739.59 → Nov 2020: 1739.59**.
reformat_min_res()	Get the **2D list** that contains the six lowest monthly average prices only as the input from find_six_highest_lowest_monthly_average_price(), reformat the date as follows, e.g., **1/2016 → Jan 2016**, and return a new list with the new date format, e.g., from **1/2016: 736.56 → Jan 2016: 736.56**.
print_info()	Get the **2D list** above as the input from get_monthly_average() and print the output as follows. Each stock price is printed to two decimal places. **Google Stock Prices Between Jan 01, 2016 and Dec 31, 2020** **Six Highest Monthly Average Prices:** **Dec 2020: 1764.70** **Nov 2020: 1739.59**

	Oct 2020: 1551.10 **Aug 2020: 1546.46** **Sep 2020: 1514.77** **Jul 2020: 1509.83** **Six Lowest Monthly Average Prices:** **Jun 2016: 712.65** **May 2016: 726.78** **Feb 2016: 729.51** **Jan 2016: 736.56** **Mar 2016: 744.12** **Jul 2016: 748.30**
main()	The function will (1) ask for the input file name: **Please enter the name of your csv file: google.csv**, and (2) call get_data_list(), get_monthly_average(), and print_info() functions in order respectively to generate the above output.

Grading Criteria:

Checkpoint:	**Possible Points**
Hierarchy Chart (pdf file)	10
Proper Naming Conventions	5
Program Documentation	5
get_data_list()	10
get_monthly_average()	10
find_six_highest_lowest_monthly_average_price()	10
reformat_max_res()	10
reformat_min_res()	10
print_info()	10
main()	10
Correct Above Sample Program Outputs	10
Total	100