DISCLAIMER: We are using actual stock market data for academic purposes only. I am NOT in any way suggesting that you invest or trade in the stock market and other financial markets. Stock market investing is RISKY, and you may lose money!

CCPROG2 MP SPECIFICATIONS PART 2 [30 points] *サルバドル・フロランテ*
AY2024 Term 2

- Part 2 covers the application of the minimum, maximum, average and search algorithms.
- The data source for Challenges 3, 4 and 5 is one of the output text files produced in Challenge #2. The file **`AC.txt`** is used as an example input file for the expected output mentioned in the remainder of this document.
- Please be reminded again that you are NOT allowed to use functions which were not discussed in class (unless specified otherwise).
- Question? Press this link and post your question in the Canvas discussion thread: Q&A on the Machine Problem (MP).

IMPORTANT NOTE: use double data type, not float, for real numbers

Philippine Stock Market Data: (some simple) Analysis

Challenge #3: Minimum & Maximum Average OHLC Price [10 pts].

An important piece of information that investors and/or traders look for are the lowest and the highest prices of a stock within a given period. For this challenge, your task is to apply the minimum and maximum algorithms to determine the MOST RECENT value of the (a) minimum average OHLC price and (b) maximum average OHLC price based on the data stored in the specified input SHD text file. The average OHLC price on a given date is computed as:

average OHLC price = (Open Price + High Price + Low Price + Close Price)/4

Example using AC.txt data: On December 27, 2019, the average OHLC price is 769.85 which is computed as

(774.94 + 782.69 + 760.89 + 760.89)/4 = 769.85

Read and follow the instructions in the accompanying skeleton file **`GROUPNUMBER-C3.c`** to produce a C program that will:

1. Input via **`scanf()`** the contents of a specified SHD text file via input redirection and store them into array data structure in the same manner as in Challenge #2
2. Determine the array indices corresponding to the MOST RECENT minimum and maximum average OHLC prices.
3. Produce an output following the format below:

```
<stock symbol>
<array index>  <date>  <minimum average OHLC price>
<array index>  <date>  <maximum average OHLC price>
```

Refer to the accompanying file **`C3-AC-EXPECTED.txt`** which shows the contents of the expected output using **`AC.txt`** as input SHD text file. The expected output is also shown in the following textbox for your immediate reference.

```
AC
962 01/21/2016 590.17
532 10/19/2017 1056.34
```

The 1st line means that the data used is for AC company. The 2nd line means that 962 is the array index where the minimum average OHLC price was found with a date of 01/21/2016 and an average OHLC price of 590.17. The 3rd line means that 532 is the array index where the maximum average OHLC price was found with a date of 10/19/2017 and an average OHLC price of 1056.34. Note that the actual number of spaces between values on the same line is not important.

Make sure that there are NO extraneous values in the output. Non-compliance in the output format will be considered as a logical error.

DELIVERABLES: Submit the following files via Canvas before the indicated submission deadline

1. C source code named as **`GROUPNUMBER-C3.c`**.
2. Your output redirected text file named as **`GROUPNUMBER-C3-AC-OUTPUT.txt`** using **`AC.txt`** as input file.
3. Output redirected text file named as **`GROUPNUMBER-C3-AC-BBTEST.txt`**[1] which shows the bbtest results between the expected result **`C3-AC-EXPECTED.TXT`** and your **`GROUPNUMBER-C3-AC-OUTPUT.txt`**.

Make sure that GROUPNUMBER is replaced with your own group number. For example, if your group number is 1, then you must upload 3 files named as **`01-C3.c`**, **`01-C3-AC-OUTPUT.txt`**, and **`01-C3-AC-BBTEST.txt.`** Single digit group numbers, i.e., 1 to 9, must have a leading zero in the filename.

Challenge #4: Compute the Simple Moving Average (SMA) of the Closing Price [10 pts.]

One of the basic technical indicators used by traders is the Simple Moving Average abbreviated as SMA. Read and understand first the idea behind SMA by referring to the links below:

- https://www.statisticshowto.com/probability-and-statistics/statistics-definitions/moving-average/
- https://www.dummies.com/personal-finance/investing/stocks-trading/how-to-calculate-simple-moving-average-in-trading/

It is also suggested that you view the following video from time 1:43 to 2:38 to learn about the SMA.

https://youtu.be/4R2CDbw4g88?t=103

If you have the time and you are interested in learning about investing or trading, view the entire video to learn how SMA is used as a technical indicator.

For this challenge, your task is to apply the average algorithm to compute the *m*-day SMA values where *m* is a positive integer parameter that represents the number of days considered in the averaging. The value of *m* varies among traders, and on the time frame considered. The more commonly used values *m = 20*, *m = 50* and *m = 200* days.

Read and follow the instructions in the accompanying skeleton file **`GROUPNUMBER-C4.c`** to produce a C program that will:

1. Input via **`scanf()`** the contents of the input file and store them into a set of array data structure, in the same manner as in Challenge #2, via input redirection
2. Compute the *m*-day SMA values of the **CLOSE** price. The SMA values must be stored in a 1D array of double data type. Moreover, the dates corresponding to the SMA values must be stored in another 1D array of string.
3. Produce as output three columns of values following the format below

```
<stock symbol>
**TEST-<m>-day-SMA**
count = <value>
<sequence>  <date>  <SMA value>
<sequence>  <date>  <SMA value>
    :         :         :
    :         :         :
    :         :         :
<sequence>  <date>  <SMA value>
```

Refer to the accompanying file **`C4-AC-EXPECTED.txt`** which shows the contents of the expected result of a 9-day SMA along with the 20-day and 50-day SMAs using **`AC.txt`** as input file.

[1] Read and follow the instructions on how to produce the **`bbtest`** file comparison result as explained in the accompanying skeleton files.

The expected PARTIAL output is also shown in the following textbox for your immediate reference.

```
AC

**TEST-9-day-SMA**
count = 1208
  1   01/19/2015   685.88
  2   01/20/2015   686.31
  3   01/21/2015   686.41
  :       :           :
  :       :           :
  1206   12/23/2019   746.26
  1207   12/26/2019   747.33
  1208   12/27/2019   748.14
```

Make sure that there are NO extraneous values in the output. Non-compliance in the output format will be considered as a logical error.

DELIVERABLES: Submit three files via Canvas before the indicated submission deadline

1. C source code named **`GROUPNUMBER-C4.c`**.
2. Output redirected text file named as **`GROUPNUMBER-C4-AC-OUTPUT.txt`** using **`AC.txt`** as input file
3. Output redirected text file of the bbtest comparison result named as **`GROUPNUMBER-C4-AC-BBTEST.txt`**.

Make sure that GROUPNUMBER is replaced with your own group number. Single digit group numbers, i.e., 1 to 9, must have a leading zero in the filename.

Challenge #5: Linear & Binary Search [10 pts.]

Apply linear and binary search algorithms to search the stock historical data for an entry that matches a specified date which is represented as a string. If the search key is found, then print the (a) array index where a matching date was found, (b) the closing price, and the (c) volume for that date.

Read and follow the instructions in the accompanying skeleton file **`GROUPNUMBER-C5.c`** to produce a C program that will:

1. Input via **`scanf()`** the contents of the input file and store them into a set of array data structure, in the same manner as in Challenge #2, via input redirection
2. Search using linear search algorithm a given search key, i.e., a date formatted as a string
3. Search using binary search algorithm a given search key, i.e., a date formatted as a string
4. Produce 5 columns of values as output following the format below

```
<symbol>

** TEST-01-LINEAR-SEARCH**
<test case #>  <date search key>  <index>  <closing price>  <volume>
<test case #>  <date search key>  <index>  <closing price>  <volume>

** TEST-02-BINARY-SEARCH**
<test case #>  <date search key>  <index>  <closing price>  <volume>
<test case #>  <date search key>  <index>  <closing price>  <volume>
```

Refer to the accompanying file **`C5-AC-EXPECTED.txt`** which shows the contents of the expected result using **`AC.txt`** as input file. The expected output is also shown in the following textbox.

```
AC

** TEST-01-LINEAR-SEARCH**
   1    12/27/2019    0    760.89     448810.00
   2    06/12/2019   -1
   3    11/09/2018  275    873.75     112620.00
   4      1/3/2017  728    704.23     205370.00
   5     09/9/2016  803    826.28     538300.00
   6     01/5/2015 1215    675.17     342780.00
   7    12/25/2025   -1

** TEST-02-BINARY-SEARCH**
   1    12/27/2019    0    760.89     448810.00
   2    06/12/2019   -1
   3    11/09/2018  275    873.75     112620.00
   4      1/3/2017  728    704.23     205370.00
   5     09/9/2016  803    826.28     538300.00
   6     01/5/2015 1215    675.17     342780.00
   7    12/25/2025   -1
```

There are 7 test cases each for linear and binary search algorithms numbered as 1 to 7 above. Consider the line of output for test case 2 (indicated in cyan background color): the search key is **"06/12/2019"**, the index is -1 which means that the search key was not found. Note that there is no stock data for this date since June 12 for any year is a holiday (in the Philippines), i.e., the stock market is closed. Consider next the line of output for test case 6 (indicated in green background color): the search key is **"01/5/2015"**, the index is 1215 which means that the search key was found. The closing price and the volume for that day were 657.17 and 342780.00 respectively.

DELIVERABLES: Submit three files via Canvas before the indicated submission deadline

1. C source code named **GROUPNUMBER-C5.c**.
2. Output redirected text file named as **GROUPNUMBER-C5-AC-OUTPUT.txt** using **AC.txt** as input file.
3. Output redirected text file of the bbtest comparison result named as **GROUPNUMBER-C5-AC-BBTEST.txt**.

Make sure that GROUPNUMBER is replaced with your own group number. Single digit group numbers, i.e., 1 to 9, must have a leading zero in the filename.

More challenges in Part 3…