

**Procedural Decomposition:****Stock Portfolio**

For this project you will modify your implementation for Project 3 with a "minor" enhancement. The output stock portfolio must be alphabetized according to the Stock name/symbol (column 1 of the input). The input will be provided in the same format as Project 3, and the output should use the same format as Project 3. In addition, all the comments from earlier projects stating that monetary values must be stored as integers still apply. Other new requirements for this project are as follows:

- Since the stock portfolio is being sorted, the input data must be stored so that it can be sorted. For Project 4, you must use Parallel Arrays to store the input stock data.
- You may use either Selection Sort, which was described in class, or Bubble Sort to sort the arrays.
- Your program must contain at least 1 Input function.
- Your program must contain at least 1 Output function.
- Your program must contain at least 2 functions that do not perform input or output. At least one of these functions must use a return code.
- Remember to add comments to your functions with a header comment for each one.
- The maximum number of input stocks in the portfolio will be 50.

**Sample input data:**

Here is a sample input file, named `StockData.txt`, for the program. The first line specifies the name of the customer, and the second specifies the customer's account number. The third line specifies the report date, followed by a blank line. The customer name, account number, and date are preceded by a single tab character. All other whitespace on these lines will be spaces. Each of the remaining lines specify the name of a particular stock and the number of shares the customer holds in the first 2 columns. In the third and the final columns are the initial and final prices for the stock, respectively. The high and low values for the stock are in the other two columns. These values are separated by single tab characters.

Customer Name	John Q Public				
Account #	432-10023-4311-89				
Report Date	06/05/2003				
MSFT	650	57.32	58.14	55.03	55.17
IBM	100	53.02	54.27	53.02	54.00
APPL	200	17.45	17.45	13.09	13.09

There is no limit on the number of lines of stock data that may be supplied; your program must terminate properly when the input stream fails. There will be no sentinel line. The input values are guaranteed to be syntactically and logically correct. The customer name, account number and date are just string data. For formatting purposes, you may assume that the customer name will be no more than 40 characters long, and the account number and date will be no longer than the ones shown above.

**Calculations:**

For each of the given stocks, you will calculate the change in total value of all the shares the customer holds, the percentage change, as well as the volatility of the stock. These values are defined as follows:

- The change in total value is the gain or loss for a given stock.
- The percentage change is the gain or loss for a stock divided by the initial price of the stock.
- The volatility is defined to be the sum of 3 values divided by the initial price of the stock. The 3 values are as follows:
  - the absolute value of the difference between the high and low stock price,
  - the absolute value of the difference between the high price and the smaller of the initial and final prices,
  - the absolute value of the difference between the low price and the larger of the initial and final prices

The comments on representing monetary values given for earlier projects still apply.

**Sample output:**

Here is a sample output file, named `Summary.txt`, which corresponds to the input data given above:

```

Programmer:  Rich Wheaton
CS 1044 Stock Portfolio

Customer:   John Q Public   432-10023-4311-89
Date:      06/05/2003

Stock      Shares      Open      Close      Change      % Change  Volatility
-----
APPL       200       17.45     13.09     -872.00     -0.250    0.75
IBM        100       53.02     54.00      98.00       0.018     0.07
MSFT       650       57.32     55.17    -1397.50    -0.038     0.15
-----
Summary                    46050.00   43878.50   -2171.50    -0.047

Winners:    1
Losers:     2

```

The first two lines just identify the programmer and project. Next there is a blank line. The next line identifies the customer name and account number, and the next is the report data. The next line contains the headers for the table.

Each line of the table gives the stock symbol, the number of shares, the initial and final stock price, the total change, the percentage change, and the volatility for a single stock. The table is bounded above and below by a single line of delimiters. The first summary line gives the overall summary data for the entire portfolio. Next there is a blank line, followed by a report on how many stocks increased in value, Winners, and how many decreased in value, Losers. Note that stocks that are unchanged are not counted.

All monetary values are reported to two decimal places. The percentage change is reported to 3 decimal places, and the volatility is reported to two decimal places.

If you have read the description of how the Curator scores your program in the *Student Guide*, you know that is important that you use the same spelling and capitalization for all the labels shown above. The horizontal spacing does not effect scoring unless you combine things that should be separate or separate things that should be combined. You are free to experiment with the horizontal layout but you should try to align the columns neatly.

**Procedural decomposition:**

Since you are using a more complex data type (parallel arrays), design which data will be stored and what data type each array will contain. Also break your overall design into smaller subtasks, which can be functions. When you define a function, make sure that you evaluate whether the function needs a return code and what parameters the function will use. Decide what functions can be coded first, such as input and output. Code and debug these functions first. Once they are working, you can trust them for the rest of your project.

**Evaluation:**

Everything that you have been told about testing in class applies here. Do not waste submissions to the Curator in testing your program! There is no point in submitting your program until you have verified that it produces correct results on the sample data files that are provided. If you waste all of your submissions because you have not tested your program adequately then you will receive a low score on this assignment. You will not be given extra submissions.

Your submitted program will be assigned a score, out of 100, based upon the runtime testing performed by the Curator System. We will also be evaluating your submission of this program for documentation style and a few good coding practices. This will result in a deduction (ideally zero) that will be applied to your score from the Curator to yield your final score for this project.

Read the *Programming Standards* page on the CS 1044 website for general guidelines. You should comment your code in the same manner as the code given for the first two programming assignments. In particular:

- You should have a header comment identifying yourself, and describing what the program does.
- Every constant and variable you declare should have a comment explaining its logical significance in the program.
- Every major block of code should have a comment describing its purpose.
- Adopt a consistent indentation style and stick to it.

Your implementation must also meet the following requirements:

- Choose descriptive identifiers when you declare a variable or constant. Avoid choosing identifiers that are entirely lower-case.
- Use named constants instead of literal constants when the constant has logical significance.
- Use C++ streams for input and output, not C-style constructs.
- Use C++ string variables to hold character data, not C-style character pointers or arrays.

Understand that the list of requirements here is not a complete repetition of the *Programming Standards* page on the course website. It is possible that requirements listed there will be applied, even if they are not listed here.

### Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to five submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and submission link can be found at:

<http://www.cs.vt.edu/curator/>

### Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:
//
// - I have not discussed the C++ language code in my program with
//   anyone other than my instructor or the teaching assistants
//   assigned to this course.
//
// - I have not used C++ language code obtained from another student,
//   or any other unauthorized source, either modified or unmodified.
//
// - If any C++ language code or documentation used in my program
//   was obtained from another source, such as a text book or course
//   notes, that has been clearly noted with a proper citation in
//   the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
//   interfere with the normal operation of the Curator System.
//
// <Student Name>
```

**Failure to include this pledge in a submission is a violation of the Honor Code.**