Presented at the DLSU Research Congress 2019
De La Salle University, Manila, Philippines
June 19 to 21, 2019

Contextualizing Programming Problems Using Philippine Stock Exchange Index Data

Florante R. Salvador*
De La Salle University, Manila
florante.salvador@dlsu.edu.ph

Abstract: We explore real-life problem of stock trading/investment using real data from 30 companies in the Philippine Stock Exhange Index as context for students to learn, design and implement basic data structures and algorithms for representing, storing and processing data in an advanced programming course. Access to real data allows us to give authentic answers to real-life questions such as "What was the lowest price of this stock"? or "How much did I earn (or lose) in my investment in this company?" This approach to specifying problems for programming exercises and projects provides student with a grounded understanding of topics with a localized context.

Key Words: Learning in context; data structures and algorithms; real-world stock historical data

1. INTRODUCTION

The trend in teaching and learning introductory programming subjects is shifting away from use of toy examples populated with concocted and synthetic data towards the processing and understanding of "diverse real world data sets" (Bart, Whitcomb, Kalura, Shaffer, and Tilevich, 2017). This is facilitated with voluminous data in the form of text, images, and statistics that have become readily available to the public for download via the internet.

The College of Computer Studies (CCS) of De La Salle University (DLSU) has been offering programming subjects since early 1980s. To the author's knowledge, there has not been any considerable effort within the last decade to incorporate real-life data in specifying programming problems and programming projects. At best, project specifications are designed to simulate real-life computing problems using synthetic data which can either be supplied by the teachers as test case data, or created by the learners themselves for testing purposes.

Progressive academic institutions in other countries are re-designing their curriculum to introduce even to first year level undergraduate students real-world data and towards data science (Anderson, Ernst, Ordonez, Pham, and Wolfman, 2014). To keep in touch with this trend, we designed

a programming project that is contextualized (Sentence and Csizmadia, 2017) using real data specifically five-year daily transaction data for the 30 companies comprising the Philippine Stock Exchange Index (PSEI).

2. PROGRAMMING SUBJECTS AT DLSU-CCS

A new curriculum that involved changes in contents of programming subjects was rolled out in CCS starting on the 1st trimester of AY2018. The first two programming subjects, coded as CCPROG1 and CCPROG2, are required core curriculum subjects for all undergraduate degree programs offered by CCS. CCPROG1 is an introductory course while CCPROG2 is the advance programming subject – which is the subject of discussion in the remainder of this paper.

In CCPROG2, students learn how to (a) represent, store, and process groups of data values stored in data structures, (b) formulate algorithms that represent solutions to specified programming problems. Learning units covered in the current syllabus include: arrays, strings, structures, and files. The C programming language is the implementation language used in implementing the algorithms.

In AY2018 Term 2, seventeen sections of CCPROG2 were offered; sixteen sections were from the DLSU Manila campus, and one section was from the DLSU Canlubang campus. Each section is composed of about 13 to 20 students.

The current CCPROG2 syllabus lists the following Learning Outcomes (LOs):

1. Analyze problem requirements by describing input specifications, processes and target output.
2. Design and implement algorithmic solutions from defined problems and requirements by applying knowledge of computing fundamentals using appropriate data structures and programming constructs including recursion.
3. Design, execute and document various classes of test cases and their corresponding results.
4. Determine and apply proper debugging techniques using programming constructs and/or computing tools.
5. Apply simple coding techniques such as inline comments, version documentation, and following coding standards for program readability.
6. Exhibit intellectual honesty, responsibility and punctuality, conforming to Christian principles.

Assessments are based on (a) exercises and assignments, (b) departmental written exams, (c) hands-on exams, (d) machine project/problem (MP), and (e) summative written final exam, each contributing 20% to the final grade. All sections followed the same syllabus with the same learning units, and were assessed using the same written and hands-on examinations. The exercises, and the MP were different among sections. This was a deliberate decision made to address possible problems on cheating and plagiarism of solutions to the MP.

The scope of the MP is larger, and requires more time and effort to accomplish compared with a simple programming exercise or a hands-on exam problem. The student would need to perform iterative activities such as think, design and implement the data structures and algorithms, code, test, debug, analyze and document the solution to a programming problem or task. Thus, the MP serves as a major evidence for an authentic assessment of the expected learning outcomes of the course. The students are given "enough time" -- at least four weeks to solve and turn in milestones of their MP.

The author was in-charge of two CCPROG2 sections with a combined size of 28 students both meeting on a Tuesday/Thursday afternoon schedule. Instead of specifying a hypothetical computing application and concocting synthetic data, a series of contextualized programming challenges were designed that will churn in numeric answers using

real-world historical daily transaction data for the 30 companies comprising the PSEI.

3. METHODOLOGY

3.1 Data Set and Data Source

There is a proliferation of data set that are now made publicly available in the internet, but a deliberate decision was made on finding data that would also have local context, that is, data that are derived from and within the Philippines. This would allow the learners to have an initial sense of familiarity. Daily transaction data from the Philippine Stock Exchange is one such possible data set. Currently, there are 270 publicly listed companies in the Philippine Stock Exchange (PSE Edge Portal, 2019) that are engaged in different business sectors including banking, electricity, energy, water, food and beverage, real estate properties, construction, mining, media, education, transportation, telecommunications, information technology. Thirty companies comprise the PSEI which serve as a proxy indicator as to the state of health of the Philippine economy.

Five-year historical daily transaction data for the PSEI covering the years 2014 to 2018 were chosen for use in the MP. To quantify the volume of data involved, there are: 30 companies, 52 trading weeks (approximately) in a year, 5 trading days in a week, 6 daily transaction data which are comprised of date, open, high, low, close (OHLC) prices and volume. Thus, for each company, there are a total of 52 x 5 x 6 = 1,560 data values for processing. Considering all the thirty companies, this means that there is a maximum of 46,800 data values. There are cases, however, that a company may have less than 1,560 transactions, for example on days that the company goes into a trading halt.

The stock historical data (SHD) are publicly available, and can be downloaded from sources such as The Wall Street Journal (WSJ). For example, stock quotes for Ayala Corporation (AC) can be accessed via http://quotes.wsj.com/PH/AC/historical-prices.

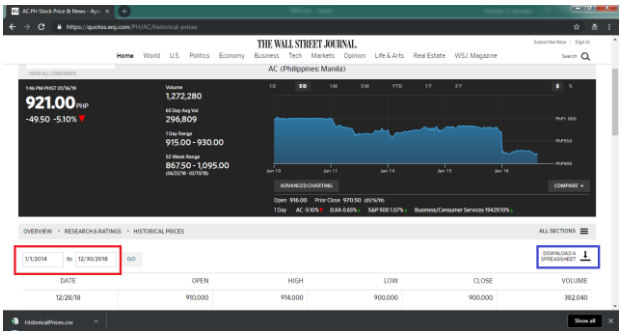

Fig. 1. Interface for downloading stock historical data from The Wall Street Journal website

3.2 Data Gathering, Data Encoding and Data Files

It was decided to let the student experience data gathering – which is not really a complicated task. Each student was given the responsibility of obtaining data for one company. Recall that there are 30 companies, and that there are 28 students. Data for AC and MBT were provided as examples of expected deliverable. As shown in Figure 1, the WSJ interface require only the following inputs: start transaction date, end transaction date and stock symbol. Thereafter, data can be downloaded as a CSV formatted file which can then be converted to a simple text file without comma delimiters.

The stock symbol and the corresponding number of daily transactions are stored on the first line of text, which are then followed by the transaction data. Following the example for AC, the abbreviated contents of the text file are:

```
AC 1215

12/28/2018  910.00  914.00  900.00  900.00  382040
12/27/2018  916.00  924.00  905.00  905.00  274520
                        :
                        :
                        :
1/3/2014    521.00  523.50  514.00  520.50  552230
1/2/2014    520.00  528.50  519.00  525.50  367200
```

where each line of transaction data stores the date, open, high, low, close prices and volume in sequence. It should be noted that data are encoded in reverse chronological order, i.e., starting from the last

transaction date in year 2018 down to the first transaction date in 2014. This ordering has an implication in searching functions as will be described later.

The thirty text files were collated, and distributed to students. The filename used is the same as the stock symbol. Thus, the 1st file was named as "AC.TXT" and the 30th file was named as "URC.TXT".

3.3 Programming Challenges

The MP specifications were framed as programming challenges labeled as C0 to C15 with tasks briefly described in Table 1. Challenge C0 is the preliminary data gathering task which is given 5 points for successful completion, while C1 to C15 are the actual programming problems worth 10 points each for a correct implementation of a solution to a task.

The challenges were formulated for the learner to demonstrate evidence of the first five learning outcomes described in section 2. All the data structures covered in the course need to be implemented and applied including arrays, strings, structures, text files and binary files. The tasks associated in the challenges indicate string processing, searching that involve dates, company names as search keys, computing for minimum and maximum values, and sorting. Space limitation prohibits detailed discussion about the challenges; the specifications document can be downloaded from https://bit.ly/2Gq8ygd

3.4 Staggered Submission and Testing

The students were provided skeleton codes with comments indicating the specific tasks to be accomplished. Morever, sample test cases and expected results were also provided. All source code deliverables were submitted via Canvas as file uploads. The submissions were staggered into four stages to allow the students to focus on a particular set of challenges (for example, C2, C3 and C4 were grouped in one set) rather than solving and submitting them all in one go. Solutions were tested during class hours via black box testing.

Table 1. List of challenges and task descriptions

Challenge	Task
C0	Gather stock historical data (SHD)
C1	Design and code your own SHD data structure
C2	Load ALL the 30 text file values into the primary memory
C3	Display the stock's daily data for a given stock code and date
C4	Display the stock's daily data for a given stock code, start and end dates
C5	Find the lowest close price for a given stock code, start and end dates
C6	Find the highest close price for a given stock code, start and end dates
C7	Find the percent change in the closing price for a given stock code, start and end dates
C8	Output into a text file a table of advancers given the start and end dates
C9	Output into a text file a table of decliners given the start and end dates
C10	Compute the N-day simple moving average of stock's close price given the stock code, start and end Dates
C11	Create a binary file of stock data
C12	Determine the number of daily data stored in the binary file given a stock code
C13	Display the daily data values stored in a binary file given the stock code
C14	Update the dates in the binary file corresponding to a given the stock code
C15	Reverse the order of historical data by date in a binary file given the stock code

The students themselves tested their own solutions based on some test cases and test scripts. Text-based feedback pinpointing logical errors in the solutions were provided to students. Feedback is crucial so as not to repeat the same type of error in subsequent challenges.

4. RESULTS AND DISCUSSION

Challenge C0 was successfully completed by all students. The only glitch that was encountered was the non-uniform encoding of dates from the

downloaded data. More specifically, some dates were encoded as 4-digit year values, while in some other companies, they were encoded by the year's last two digits only (for example, 15 for year 2015). To minimize possible complications, the data files were pre-processed such that all dates conform to a 4-digit year format before they were distributed to the students. In hindsight, distributing the original data would have been a good opportunity for the students to be initially exposed to the raw nature of data sets which need pre-processing or data cleaning.

Challenge C1 was a crucial task and affected how data will be accessed in all the succeeding tasks. C1 specifically targeted the data structure aspect mentioned in Learning Outcome 2. The submitted data structure for representing and storing all the stock historical data (in the main memory) was in the form of an array of structures. Each structure contains a stock symbol, the number of transactions in five years, and another array of structure containing the daily historical data comprised of date, open, high, low, close prices values and volume. It is interesting to note that one advanced student designed and implemented data structure and algorithms using dynamic memory allocation – which was a topic not included in CCPROG2.

Submission of the solutions for the programming challenges with function definitions as deliverables were clustered and solved in sequence based on the list of topics in the syllabus. Specifically, challenges C2 to C7 focused on arrays, strings and structures, C8 to C10 required writing results into text files, and C11 to C15 involved reading, writing and modifying data in binary files. Correct solutions involved proper formulation of algorithms that involved searching (unique keys and range of key values), computing for minimum and maximum values and sorting.

The students' average performance for these challenges are shown in Figure 2. In general, the students performed quite well. High ratings, i.e., at least a score of 9.0, were obtained in 10 out of the 14 challenges. However, a low rating of 6.04 is seen in C4. It should be noted that C3 and C4 both involved formulating a search algorithm. The difference is that C3 involves only a single search date key value (for example, display the stock data on 10/24/2016) while C4 involves two date values, a start date and an end date (for example, display one month of stock data covering the period 4/1/2017 to 4/30/2017). The

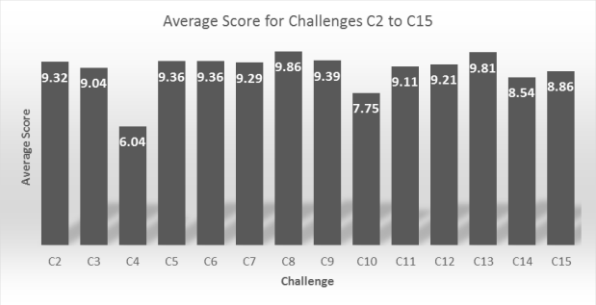

Fig. 2. Average score for each programming challenge

two tasks seem to be almost the same. In fact, C4 can be implemented to call the function definition for C3. However, the results proved that C4 was quite a difficult task. Upon inspection of the submitted codes, there were mainly two causes of logical errors identified. The search algorithm

a. did not account for the fact that the dates were stored in the text file in reversed chronological order (refer back to subsection 3.2)
b. failed because of incorrect processing of days without trading (note that 4/1/2017 was a Saturday and 4/30/2017 was a Sunday)

which caused array indices to go out of bounds resulting either in an incorrect output or run-time error.

The errors identified and the accompanying feedback provided to students helped in the remediation. The students were able to fix their searching algorithms in the succeeding challenges, in particular C5 to C10, which all involved searching within a certain range of dates. This is the primary advantage of having staggered submission and testing (see subsection 3.4).

The other low assessment score was 7.75 in C10 which required the computation of simple moving average of the closing price for N-days. The common logical error observed in the submitted solutions was due to

a. misunderstanding of the formula/process for computing the simple moving average, and/or
b. incorrect conversion of the formula to C programming language codes.

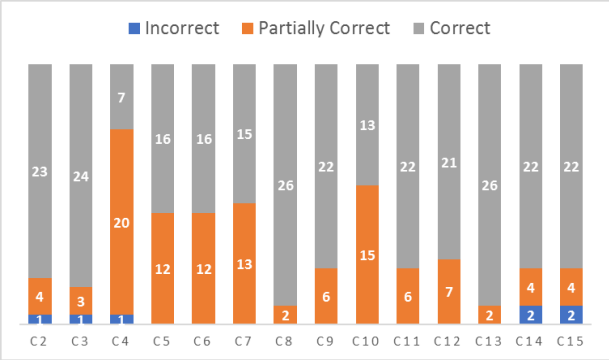

Fig. 3. Frequency count of incorrect, partially correct and correct solutions for each programming challenge

A frequency count of solutions classified as incorrect (i.e., score is 0), partially correct, and correct (i.e., score is 10) for each challenge is visualized in Figure 3. It should be noted that the scores in many of the partially correct solutions were in the higher range of scores, i.e., 8 to 9. For example, even if there were only 12 partially correct as opposed to 16 correct solutions for challenges C5 and C6, the average score was still high at 9.36.

The C4 frequency count likewise indicates that it was the most difficult challenge with only 7 correct solutions against 20 incorrect solutions with many of the scores lying in the middle to lower values, i.e., 5 or less. As for those counted as incorrect solutions, the cause was due to incorrect file submission (C2 to C4), and not implementing the required functions (C14 and C15).

Overall, the results indicate that there was incremental improvement in the students' understanding of the topics and programming skill, part of which can attributed to learning from mistakes made in previous challenges.

4. CONCLUSION AND FUTURE WORK

Instead of specifying a machine problem that only attempts to mimic real-life problems with synthetic data, we explored the use of real data from the PSEI for students to learn, design and implement basic data structures and algorithms for representing, storing and processing data in an advanced programming course. Access to real data enabled the students to formulate algorithms and code programs that provide realistic answers to questions such as "What was the lowest price of this stock"? or "How much did I earn (or lose) in my investment in this company?"

Informal verbal feedback from students indicated that the challenges were "moderate to difficult" but in general solvable. We still need to determine via survey if the chosen context actually engaged the student and if the MP was able to maintain the student's interest in learning the subject.

5. REFERENCES

Anderson, R., Ernst, M., Ordonez, R., Pham, P. & Wolfman, S. (2014). Introductory Programming Meets the Real World: Using Real Problems and Data in CS1. In Proceedings of the ACM SIGCSE '14, pages 465-466.

Bart, A. C., Whitcomb, R., Kafura, D., Shaffer, C. & Tilevich, E. (2017). Computing with CORGIS: diverse, real-world data sets for introductory computing. In Proceedings of the ACM SIGCSE '17, pages 57-62.

PSE Edge Portal (2019). http://edge.pse.com.ph/

Sentence, S., & Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspectives. Education and Information Technologies, volume 2, number 2, pages 469-495.