USING INTERVIEW QUESTIONS AS SHORT-TERM PROGRAMMING ASSIGNMENTS IN CS2

Timothy Urness
Department of Mathematics and Computer Science
Drake University
Des Moines, IA 50311
515-271-2118
timothy.urness@drake.edu

ABSTRACT

In this paper, we describe our experiences with using technical interview questions as the basis for programming assignments in a CS2 course. Several books have been published that chronicle popular questions that have been asked on technical interviews at companies such as Microsoft, Google, Apple, Facebook, and Amazon. The books are written as a service to professionals interviewing for a job, but they also serve as an excellent set of short exemplar questions from concepts that are typically covered in an introductory programming sequence. We found that the programming assignments based on interview questions were particularly motivating for students. The interview questions, given in short-term programming assignments throughout the semester, resulted in an increased performance on midterm exams and final exams when compared to the performance of students in a section that utilized standard programming assignments.

INTRODUCTION

We have recently noticed students preparing for a technical interview by studying a book designed to prepare applicants for a variety of possible questions they might encounter. Upon closer evaluation, we discovered several books [2, 7, 8] that give advice for the preparation for technical interviews at companies such as Microsoft, Google, Apple, Facebook, and Amazon. These resources chronicle popular questions that have been asked by interviewers. The questions often probe an applicants' understanding in algorithm analysis and basic data structures (e.g. arrays, stacks, linked lists, queues, hash tables), which is typically much of the content of CS1 and CS2 courses.

A standard section of CS2 at Drake University, a small liberal-arts college, would traditionally entail several standard, long-term programming projects that are large in scope, require many lines of code, and would have deadlines of approximately one to two weeks. In the updated interview question assignments section, students were assigned smaller projects throughout the semester that were due before the following class meeting. The smaller projects were largely inspired by questions that were asked at interviews for large technology companies, as published in various technical interview preparation books.

Our hypothesis was that students could benefit from the rapid interview question assignments due to the increased, intensive practice required of the short-term assignments. We also believed that the fact that these assignments were inspired from "real world" interview questions would help inspire and motivate the

students. We also wanted to measure if the absence of long-range software projects affected students' understanding of software engineering principles and concepts that would be introduced and discussed in class, but not practiced until later courses in the curriculum. In addition to anecdotal observations, we analyzed these sets of skills and abilities through a final exam that was common to both sets of students. The final exam included questions that allowed for the assessment of fundamental, small-scale data structure comprehension and development as well as large-scale software development questions.

RELATED WORK

Numerous studies have shown the importance of programming assignments in the introductory CS1 and CS2 sequence. Several studies have suggested the importance of making assignments "meaningful" or of a practical usefulness for student engagement and attracting students to computer science [1, 6, 11].

Nick Parlante, from Stanford University, annually hosts a "nifty assignments" session at the SIGCSE conference that showcases several assignments which have been proven to be particularly popular, useful, illustrative and adoptable for other professors [10]. The publication of these assignments is a tremendous service for faculty looking for proven assignments that have engaged students on a variety of topics, typically in CS1 or CS2.

Many studies have utilized meta-data that has been automatically collected in student assignment submissions. Data such as when the students first submit solutions, what kinds of compilation errors students experience, and how often students submit can give clear indications as to the effective and ineffective behaviors of student programmers. Many of these studies reinforce the intuition of professors: starting assignments sooner often results in better grades and fewer frustrations [3, 4]. Also, students tend to underestimate how long assignments will take [9].

METHODOLOGY

The CS2 course at Drake University focuses on the use and implementation of various data structures: arrays, linked lists, binary trees, stacks, queues, hash tables, and heaps. Each class meeting typically involves a lecture by the professor in addition to in-class, hands-on student programming exercises on individual laptops. Classes are taught twice a week; each class meeting lasts 75 minutes. The course is evaluated with two midterm exams and a comprehensive final exam.

The session of the course which utilized the traditional, long-term assignments consisted of 33 students and was taught in the spring of 2015. Nine of the students were female (27.3%). The assignments were taken from various sources, including several nifty assignments [10]. The short-term interview question assignments section of the course consisted of 24 students and was taught in the fall of 2015. Eight of the students were female (33.3%). The assignments were taken from various sources, including the textbook [5] and technical interview questions encountered by students as well as technical interview resources [2, 7, 8]. Each assignment was due prior to the next class period.

RESULTS AND ANALYSIS

Exam Averages

Initially, the students in the long-term assignment section appeared to be better equipped for the course material than their counterparts in the short-term assignment section of the course. Anecdotally, the long-term assignment section students appeared to be extraordinarily interested and engaged in the material. This was evidenced by the discrepancy in the results of the first exam, which was administered roughly halfway through the semester (88.73% vs. 83.85%).

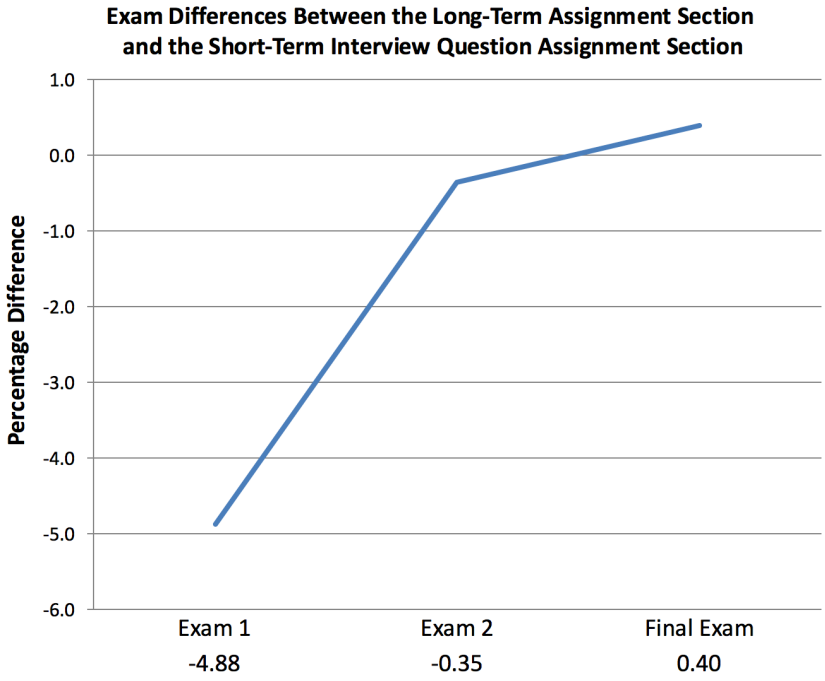

Figure 1: Relative exam differences between the two sections. The interview question assignment section average subtracted from the long-term assignment average highlights the difference between the sections.

However, the students that were enrolled in the interview question assignment section appeared invigorated by the continual challenges of the short-term, interview question assignments and were undeterred by early challenges. These students thrived during the semester, particularly bolstered by the practical content of the assignments, which were inspired by technical interview questions. The effects of the increased practice generated by the frequent short-term assignments are seen by the second midterm in which the score were remarkably similar (90.0% vs. 90.4%). By the final exam, the short-term assignment group surpassed the long-term assignment group in average exam score (85.2% vs. 85.6%).

Final Exam Questions

Table 1 displays a selection of the exam questions that highlight the most significant differences between the two sections.

#	Final exam question
1	A class is similar to a structure in that they define an abstract data type. How are classes and structures different?
2	What is a function prototype? Why is it used?
3	How is a queue different than a linked list? How are they similar?
4	What are the advantages of using a STL vector over a standard array? List two.
5	How is a binary search tree different than a complete binary tree?
6	What advantages does a hash table have over an array? Explain why.
7	In Computer Science, what is the "big-Oh" notation used for and why is it useful? What is the big-Oh running time of the following algorithm? (algorithm listed – merge sort)
8	Given a string as a parameter, write a method that determines if the string is a palindrome (is the same forward and backwards – e.g. otto, mom, dad, racecar). The function should return either true or false.
9	Write an example of recursion**.**
10	Given the code **int num = 25;** **int *ptr;** Finish the code, using the variable ptr, to print out the value 25.
11	Given the following picture of memory, what will be output when the code is run?
12	Implement a **Set** data type that will work for storing a mathematical set of integers.
13	Suppose you were asked to write the software that would handle an online voting system for a college campus. Describe (and/or diagram) the classes how you would use. Also explicitly indicate how you would use inheritance.

Table 1: A selection of final exam questions.

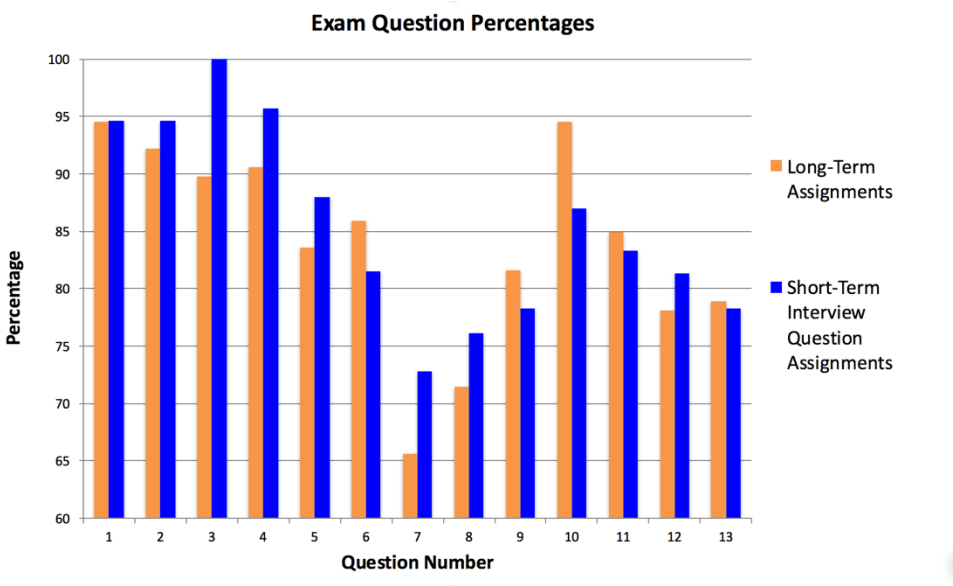

Figure 2: Average percentage of exam questions of the standard assignments section and interview question assignments section

Interview Question Assignments Advantages

Of particular interest are questions in which the interview question assignment section performed better than the standard assignment section.

Questions 3, 4, 5, and 8 involve the properties of data structures and the details of how data structures differ. The interview question assignment section was able to cover more of these kinds of question with assignments. As a result, the students had individually practiced on these topics and were not only limited to classroom coverage and in-class exercises of the materials. In general, the short-term interview question assignments section was more focused on implementation details of a wider variety of data structures than the long-term assignment section allowed, as evidenced by the improved performance on these questions.

Long-Term Assignments Advantages

Questions in which the long-term assignment section performed better than the short-term section include question 6, which asks about the hash table data structure. The long-term section had a specific assignment in which the students thoroughly implemented the hash table in the context of an assignment. Similarly, question 9 (recursion) was reinforced by an assignment. This group also performed well on the questions regarding pointers (question 10). This concept was covered early in the semester, and the students in the standard assignment section, on average, showed a high aptitude for this material, as evidenced by the first exam results.

The Length of Programming Assignments

When presented with a long-term programming assignment in an introductory course, students have a few things immediately working against them. First, starting on the assignment early does not obviously nor immediately reward students. While some students have certainly taken the advice from professors to start on assignments early, there exists sufficient data that many students do procrastinate working on assignments to a point where it negatively affects their performance in the course [3, 4, 9]. Studies have shown that students who do not start working until the day before an assignment is due perform significantly poorer than those who start two or more days before the assignment is due. A second detriment to the long-term assignment is that students often underestimate the amount of time programming projects will require [9].

Thus, the long-term assignment can form a trap: starting on the assignment early does not obviously nor immediately reward students. Once behind, students may not realize they are behind as they don't accurately estimate the time that will be required to complete the assignment.

CONCLUSIONS

We believe that students benefit from developing a habit of practicing to program on a regular basis. Our experience is that requiring assignments with short deadlines that draw upon technical interview questions motivate students to develop consistent programming habits that result in increased performance on midterm and final exams. The increased practice and focus on implementation, as well as the breath of coverage on the assignments, ultimately benefited the short-term assignment students. The regularity of the assignments didn't allow for long-term procrastination.

However, the importance of long-term programming assignments cannot be understated. In addition to the conceptual understanding and implementation of basic data structures, students should have practice utilizing data structures in complicated, involved solutions that require the development of sound, software engineering practices. It is our opinion that CS2 is an excellent place to establish the fundamentals of the data structures using short interview-like programming assignments. Long-term assignments can effectively be used in upper-division courses to develop a deeper understanding and practice using the tools introduced in CS2.

In summary, we feel that the best practice for the instruction of students learning programming in an introductory sequence is to utilize short-term deadlines that properly incentivize students to get started working on assignments with regularity sooner rather than later. Long-term, "nifty" assignments are excellent resources and could be most effective with shorter-term midpoint deadlines that ensure students do not underestimate the necessary time for completion.

REFERENCES

[1] Anderson, R.,E., Ernst, M.,D., Ordóñez, R., Pham, P., Tribelhorn, B., A Data Programming CS1 Course*. Proceedings of the 46th ACM Technical Symposium on Computer Science Education (SIGCSE '15)*, 150-155, 2015.

[2] Aziz, A., Lee, T. H., Prakash, A., *Elements of Programming Interviews: The Insiders' Guide*, CreateSpace Independent Publishing, 2012.

[3] Edwards, S.H., Snyder, J., Pérez-Quiñones, M.A., Allevato, A., Kim D., Tretola, B., Comparing effective and ineffective behaviors of student programmers. *Proceedings of the fifth international workshop on Computing education research workshop (ICER '09)*, 3-14, 2009.

[4] Fenwick, Jr. J.B., Norris, C., Barry, F.E., Rountree, J., Spicer, C.J., Cheek, S.D., Another look at the behaviors of novice programmers. *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09)*, 296-300, 2009.

[5] Gaddis, T., *Starting Out with C++ from Control Structures to Objects*. Upper Saddle River, NJ., Pearson Education, Inc., 2015.

[6] Layman, L., Williams, L., Slaten, K., Note to self: make assignments meaningful. *Proceedings of the 38th SIGCSE technical symposium on Computer science education (SIGCSE '07),* 459-463, 2007.

[7] McDowell, G. L., *Cracking the Coding Interview: 150 Programming Questions and Solutions, 5th Edition,* Palo Alto, CA., CareerCup LLC, 2013.

[8] Mongan, J., Suojanen, N., Giguère, E., *Programming Interviews Exposed: Secrets to Landing Your Next Job.* Indianapolis, IN: John Wiley & Sons, 2012.

[9] Murphy, C., Kaiser, G., Loveland, K., and Hasan, S., 2009. Retina: helping students and instructors based on observed programming activities. *Proceedings of the 40th ACM technical symposium on Computer science education (SIGCSE '09)*, 178-182, 2009.

[10] Parlante, N. Nifty Assignments, http://nifty.stanford.edu/, 2016.

[11] Stevenson, D. E., Wagner, P., J., 2006. Developing real-world programming assignments for CS1. *Proceedings of the 11th annual SIGCSE conference on Innovation and technology in computer science education (ITICSE '06),* 158-162, 2006.