Including Computer Systems Assignments in Introductory Programming Courses

Christine F. Reilly
Computer Science Department
Skidmore College
Saratoga Springs, New York, USA
creilly@skidmore.edu

Aarathi Prasad
Computer Science Department
Skidmore College
Saratoga Springs, New York, USA
aprasad@skidmore.edu

***Abstract*—This Research to Practice Work In Progress paper describes initial work on designing and deploying programming assignments that focus on computer systems topics in introductory computer science courses. This work builds upon the body of prior work that has found that using real-world problems in introductory courses improves student engagement and helps students in the process of developing their professional identity. The setting for this research is a liberal arts college where the major has been designed to support the liberal arts mission of providing students with opportunities to pursue a variety of interests. By including computer systems topics in the introductory courses we are able to increase breadth while maintaining a relatively small number of required courses for the major. The four assignments described in this paper focus on the computer systems areas of operating systems, networks, and database systems. Informal feedback and instructor reflection from the initial deployment of these assignments indicates that students are engaged with these assignments. Future work includes a formal assessment of this approach.**

***Index Terms*—computer science, introductory course, assignments, computer systems**

I. Introduction

In this Research to Practice Work-in-Progress paper we describe initial work on designing and deploying programming assignments that focus on computer systems topics in introductory computer science courses. The topics of these assignments are from the computer systems areas of operating systems, networks, and database systems. Incorporating assignments that focus on computer systems topics builds upon the body of prior work that has found that using real-world problems in introductory courses improves student engagement and helps students with professional identity formation.

There are two primary motivations for bringing these computer systems topics into the introductory course assignments. First, these assignments are deployed at a liberal arts college where the computer science major has a relatively small number of required classes. The computer systems electives are not necessarily offered every year, and students may choose to take other computer science topics as their elective courses. By bringing computer systems topics into the introductory course assignments, the students have an opportunity to gain exposure to these topics regardless of the elective courses they take later in their studies. Including computer systems topics in the introductory course assignments is a way to increase breadth in the major while maintaining a small number of required courses and supporting the liberal arts mission of providing students with the opportunity to pursue a variety of interests during their college studies.

The second motivation is an observation that students, and the general public, often think that computer science is programming and that computer scientists' work is to write application programs. Including systems topics assignments in the introductory courses provides computer science majors, as well as students from other majors who are taking the programming courses, with examples of the types of problems that computer scientists focus on.

A goal of presenting this work as a work-in-progress paper is to obtain feedback on this initial work and to seek additional collaborators. Long term goals of this project include expanding the range of assignment topics to include areas of computer science beyond computer systems. With collaborators who have a variety of expertise, we will create a library of assignments that focus on a broad area of computer science specialties, providing a useful resource to the community. We also welcome collaborators with experience in education research who are interested in formally evaluating the outcomes of these assignments.

II. Related Work

Professors who teach introductory computer science courses have long been encouraged to design engaging assignments that focus on real world problems. One approach for designing engaging assignments is to use games as the topic of the assignment [1]–[4]. Another approach is to incorporate more advanced computing concepts, such as network programming [5], into introductory course assignments. There are repositories of engaging assignments that faculty can reference [6], including ones that have been peer-reviewed and meet best practices for student engagement [7].

Research on professional identity formation has found that coursework is one of the ways that students learn about careers in the computing profession. Because students typically form their professional identity around the third year of a four year undergraduate program, it is important to provide opportunities for students to learn about different areas of computing and career possibilities during the first two years of the program [8].

In the early undergraduate years, many students view computer science as a field that focuses on creating smaller digital artifacts, and students who do not feel that they fit in with this focus often leave the program [9]. Having discipline-specific courses early in the program combined with explanations of why students are learning a topic and how they will use this knowledge in their future work is helpful for encouraging professional identity formation in computing disciplines [10]. Diversity in computer science may be improved by programs providing guided opportunities for students to learn about and value the breadth of types of work that is done by computing professionals, especially when these learning opportunities are placed in the early years of the program [9].

Computer science programs at liberal arts colleges face a variety of challenges related to providing breadth and depth in the major [11], [12]. Due to having a small number of faculty members, these programs typically have limited course offerings. The computer science course offerings may be further limited when departments are also offering courses that fulfill the college's liberal arts requirements such as first year seminar courses, writing courses, and other broader liberal arts courses. Faculty who teach introductory computer science courses at a liberal arts college often need to be creative in designing a course that meets the needs of students who intend to major in computer science as well as students from other majors who take one or two computer science courses as part of the liberal arts philosophy of exploring a variety of areas in their studies [13].

III. Context

This study is conducted at a liberal arts college in the Northeast United States. The college enrolls 2,500 undergraduate students. In recent years, there have been approximately 20 students per graduating class who major in computer science. The Computer Science department has five full time faculty members: two tenured faculty, two tenure-track faculty, and one lecturer.

The Computer Science (CS) department and major are similar to the median liberal arts computer science program as described in [11]. 40% of the credit hours required for graduation are courses that are required for the Computer Science major. The CS department has designed a major that has a relatively small number of required courses so that students are able to take advantage of the opportunities that the liberal arts college provides for choosing to focus their studies on their own interests. Many CS majors participate in study abroad, have a second major, or have one or more minors. Some CS majors choose to take additional CS electives. At this college, students must declare a major by the middle of the second semester of their Sophomore year. Students who declare a computer science major are expected to have completed Introduction to Computer Science I (CS1) and either have completed or be currently enrolled in Introduction to Computer Science II (CS2) when they declare a major. It is possible for students to declare a second major at any point up to the first semester of their senior year.

This CS department faces the typical challenges that CS programs at liberal arts colleges have in offering a major that provides sufficient breadth and depth [11], [12]. In addition to having a relatively small number of courses required for the major, our course offerings are limited by the small size of our faculty. We address these challenges by creatively designing courses with learning outcomes that would be found in multiple classes in a bigger department. For example, we include software design in our Computer Organization course through a semester-long Java programming project where students build a simulated computer [14].

The work discussed in the current paper is another example of using a single course to serve multiple purposes. We use the assignments in the CS1 and CS2 courses to introduce concepts from computer systems, and will expand the assignment topics in the future to encompass additional areas of computer science. This approach provides students with exposure to the breadth of computer science early in the major. As was discussed in Section II, learning about the breadth of computer science during the early years of an undergraduate program is expected to assist students with professional identity formation, provide information students can use when selecting technical electives that match their interests, and demonstrate the connections between courses in the major [8]–[10], [13].

Designing assignments for CS1 and CS2 courses that focus on computer systems topics diverges from much of the prior work on using assignments to promote engagement in introductory courses and retention in the major. The prior work, as discussed in Section II mostly focuses on assignment topics that students can relate to in their own lives. As part of our future work, we will evaluate whether including computer systems topic assignments has an impact on students' interest in majoring in Computer Science. We note that there are other assignments in these courses that focus on problems outside of computer science. Additionally, if the computer systems topic assignments are assisting with the process of professional identity formation then we are likely to see a positive impact from these assignments.

IV. Examples of Assignments

In this section we provide four examples of computer systems topic assignments we have used in introductory computer science courses. Table I provides a summary of these assignments including the course learning goals demonstrated by each assignment. These courses were offered at a liberal arts college where there are typically between 15 and 20 students enrolled in the course. There are two introductory courses: Introduction to Computer Science I (CS1) and Introduction to Computer Science II (CS2). The CS1 course focuses on programming concepts including conditional execution, iteration, functions, recursion, simple data structures such as lists, and an introduction to object oriented programming. The CS2 course has a prerequisite of the CS1 course and focuses on data structures and the algorithms that operate on those data structures as well as on object oriented programming. The

data structures discussed in the CS2 course include linked lists, stacks, queues, priority queues, heaps, graphs, trees, and hash tables. We note that the department is currently transitioning to a new curriculum. The CS1 class has switched from using Java to the Python programming language, and the CS2 course is splitting into into two courses. The CS1 assignments described in this paper were deployed in the new version of the class that is taught in Python, while the CS2 assignments were deployed in the old version of the class that was taught in Java and had a prerequisite of the CS1 Java class. In future work we will discuss how these CS2 assignments are transitioned into the new curriculum.

A. Database Query Processing and Data Science

In the seventh and eighth out of nine programming assignments in the CS1 class, students create a simple database using a list of lists (essentially a two-dimensional array) along with functions that operate on this structure. The programming skills that are the focus of these assignments are using the list, dictionary, and tuple data structures in Python. These assignments also incorporate concepts from throughout the semester including iteration, recursion, and writing functions.

In Program 7, students write functions to load data from a file into the list structure and to print the list structure. They also write functions that mimic typical database operations of selecting rows where one field in the row matches a specified value, and projecting specific columns. Other functions perform tasks typical of data science applications by asking students to create a dictionary (hash table structure that contains key value pairs) where the key is the top level domain name as extracted from the email addresses in the database and the value is the count of occurrences of this name, and then to print this dictionary sorted by the the value.

In Program 8, students write two functions for selecting a single row from the database, under the assumption that the data for the select criteria has a unique value in each row. One function uses the linear search algorithm, and the other function uses the binary search algorithm. Both functions return a tuple data structure that contains the desired row along with the count of number of rows that are examined before the desired row was found. Note that the code for the recursive binary search is provided to the students. Their task is to add code to the recursive binary search function to count the number of rows that are examined. The provided test programs demonstrate the better performance of binary search by showing the number of rows examined when searching for the same value using both functions. A third test program illustrates a common computer systems performance comparison process of calling the functions many times and finding the average number of rows examined by each of the functions.

We received informal feedback from one of the peer tutors who commented that the CS1 students were working on a "cool data science" assignment. This informal feedback indicates that this assignment topic is meeting our goal of providing students in the CS1 course with examples of real work done by computing professionals.

B. Memory Free List

The second of five programming assignments in the CS2 class focuses on linked lists. For this assignment, students wrote a Java program using a doubly and circularly linked list to simulate the memory free list that an operating system uses to keep track of available areas of computer memory. We chose a doubly and circularly linked list so that the assignment requires students to implement a linked list structure that has differences from the linked lists that are discussed in class and in the textbook.

In this assignment, students create the FreeList class based on the provided specifications. The doubly and circularly linked list is represented as a collection of nodes where each node has a reference to its previous and next nodes. Instead of having a reference to the head or tail of the list, the class contains a data member that refers to the node in the list where memory was most recently allocated. The class includes public methods for removing memory from the free list when the operating system allocates memory, and for adding memory to the free list when the operating system frees memory. The class also has the following common object oriented programming methods: default constructor, parameterized constructor, and a method that returns a string representation of the object. Another required public method returns the number of items in the list, providing a similar interface as other collection classes in Java. Students are provided with a class that represents a single node in the linked list, and with a class for the data that is associated with each free slot in memory. Test programs are also provided. This is a large and complex programming assignment, and the assignment instructions provide a list of steps to guide students through incrementally building and testing the code.

This assignment is successful in terms of the amount of understanding it develops about the linked list concept. In the allocate method, students must write code for removing a node from the list. The free method requires students to coalesce adjacent areas of memory by combining nodes in the list. These operations provide students with many opportunities to practice adjusting node references to the previous and next node. The free method and the size method require students to write code that traverses the list.

Informal feedback from students indicated that the top performing students in the class found this to be an interesting and challenging programming assignment. Some of the lower performing students in the class seemed to be overwhelmed by the difficulty of this assignment. One change we are considering for the future is specifying different names for the methods that modify the free list. Some students found the names of these methods confusing because the names are related to the operating system's actions not to the actions on the list: the allocate method removes memory from the free list to allocate to the operating system, and the free method adds memory to the free list when memory is freed by the operating system. We will also modify the test code by adding smaller scale tests that provide additional scaffolding for students who

TABLE I
EXAMPLES OF COMPUTER SYSTEMS PROGRAMMING ASSIGNMENTS IN INTRODUCTORY CS COURSES

Assignment	Course	Course Learning Goals
Database Query Processing and Data Science	CS1	Use of simple data structures (lists, dictionary, tuples); Understanding of linear and binary search algorithms.
Memory Free List	CS2	Implementation of linked lists; Object oriented programming.
Process Scheduler	CS2	Implement priority queue as array of queues; Appreciate how data structures can be used together; Object oriented programming.
Routing Algorithms	CS2	Implement graphs, hash table, and heaps; Understand the use of multiple data structures to solve a problem; Choose the most efficient data structure for a specific problem.

are struggling with the size and complexity of this assignment.

C. Priority Scheduler

The fourth of five programming assignments in the CS2 class focuses on priority queues. For this assignment, students use an array of queues to simulate a priority job scheduler that an operating system uses to choose a process to run.

In this assignment, students create a Process class consisting of three fields: name, runtime and priority. They also write a Scheduler class to simulate the scheduling algorithm. The scheduling algorithm in this assignment is implemented using five public methods. An add method takes a Process object as a parameter and places it at the scheduler-defined highest priority. The run method uses three integer variables that act as timers to keep track of boost time, the time slice, and the runtime of the process. If the runtime reaches zero before the time slice, it is removed from the scheduler, otherwise the age method is invoked to move the Process object to a different queue. Students also implement a boost method, which is invoked when the boost timer reaches zero.

In terms of course learning goals, this assignment was successful since the students revisited the concepts of arrays that they learned in the CS1 class and queues that they learned earlier in this CS2 class. They also expanded their object oriented programming skills by writing a program that utilizes multiple Java classes and experienced how one object can use an object of a different class by calling its public methods.

Informal feedback indicated that the top students found the assignment to be interesting, while others found the assignment to be confusing, especially the use of integer variables as timers. In the future, we will improve the test program that is provided to students by adding smaller scale tests that will provide additional scaffolding to assist students with managing the complexity of this assignment.

D. Routing Algorithm

The last of five programming assignments in the CS2 class focuses on graph algorithms. For this assignment, students implement shortest path algorithms for both weighted and unweighted graphs. Routing algorithms are a computer systems topic that illustrates these concepts.

In this assignment, students create a Graph class and a NetworkNode class. The NetworkNode class contains a hash table that stores the next-hop nodes for every other node in the network. Given a destination node the hash table provides the next node to send the message to. The students write code using Dijkstra's algorithm to populate the next hop table in the NetworkNode object.

This assignment meets the learning goal of a final assignment in the CS2 course that demonstrates the use of three different data structures: a graph, a heap used as part of the implementation of Dijkstra's algorithm, and a hash table that stores the next hop values. By working with these three complex data structures, students gain experience with choosing the most efficient data structure for a given scenario. Additionally, this assignment introduces the concept of routing algorithms. The assignment information discusses how routing algorithms are utilized for multiple applications including routers and hosts on the Internet, and in computer applications such as location-based services.

V. DISCUSSION AND CONCLUSION

This work in progress paper presented the motivation for including computer systems topic assignments in introductory computer science courses and described four assignments that we have deployed in our courses. These assignments are a novel approach for adding breadth in the early part of a CS major. Based on informal feedback from students and our reflections, these assignments are meeting the course learning goals. One lesson we have learned from the initial deployment of these assignments is that the complexity of the computer systems topics requires us to include small scale tests that provide scaffolding to assist students in understanding how to accomplish the tasks required in these assignments.

The next step in this project is to design a formal assessment that can measure whether these computer system topic assignments are assisting with professional identity formation. The prior work on computer science professional identity formation has been conducted at larger universities. An interesting research question is how professional identity formation at our liberal arts college compares with that of students at larger universities. Another interesting research question is the impact of these computer systems topic assignments on the engagement and retention of students from populations that are underrepresented in computing.

ACKNOWLEDGMENTS

Thank you to the Skidmore College students in our CS106 and CS206 classes who provided helpful feedback about these assignments.

REFERENCES

[1] J. D. Bayliss and S. Strout, "Games as a "flavor" of CS1," in *SIGCSE 2006*, 2006.

[2] P. Drake and K. Sung, "Teaching introductory programming with popular board games," in *SIGCSE 2011*, 2011.

[3] C. F. Reilly and N. De La Mora, "The impact of real-world topic labs on student performance in CS1," in *2012 Frontiers in Education Conference Proceedings*, 2012.

[4] K. Sung, M. Panitz, S. Wallace, R. Anderson, and J. Nordlinger, "Game–themed programming assignments: The faculty perspective," in *SIGCSE 2008*, 2008.

[5] M. H. Goldwasser and D. Letscher, "Introducing network programming into a CS1 course," *SIGCSE Bull.*, vol. 39, no. 3, pp. 19–22, jun 2007.

[6] N. Parlante, J. Zelenski, A. A. de Freitas, T. B. Weingart, K. Schwarz, B. Stephenson, and S. Bitner, "Nifty assignments," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 2021.

[7] A. E. Monge, C. L. Fadjo, B. A. Quinn, and L. J. Barker, "EngageCSEdu: Engaging and retaining CS1 and CS2 students," *ACM Inroads*, vol. 6, no. 1, pp. 6–11, Feb 2015.

[8] A. Kapoor and C. Gardner-McCune, "Understanding CS undergraduate students' professional identity through the lens of their professional development," in *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. Association for Computing Machinery, 2019, pp. 9–15.

[9] A.-K. Peters, "Students' experience of participation in a discipline—a longitudinal study of computer science and IT engineering students," *ACM Transactions on Computing Education*, vol. 19, no. 1, Sep 2018.

[10] G. M. Lundberg and I. J. Ness, "First year students' imagination of future employment: Identity as an important employability aspect," in *Proceedings of the 9th Computer Science Education Research Conference*. Association for Computing Machinery, 2020.

[11] D. Baldwin, A. Holland-Minkley, and G. Braught, "Report of the SIGCSE committee on computing education in liberal arts colleges," *ACM Inroads*, vol. 10, no. 2, 2019.

[12] S. G. M. Koo, "Computer science curriculum in a liberal arts setting: Case studies at the University of San Diego," in *Proceedings of IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE) 2012*, 2012.

[13] A. Brady, P. Cutter, and K. Schultz, "Benefits of a CS0 course in liberal arts colleges," *Journal of Computing Sciences in Colleges*, vol. 20, no. 1, October 2004.

[14] C. F. Reilly and J. A. Swanson, "A case study in constructivist pedagogy in a computer organization course," in *2019 IEEE Frontiers in Education Conference (FIE)*, October 2019.