The Journal of Computing Sciences in Colleges

Papers of the 33th Annual CCSC South Central Conference

April 8th, 2022
The University of Texas at Dallas
Richardson, TX

Volume 37, Number 7 **April 2022**

The Journal of Computing Sciences in Colleges (ISSN 1937-4771 print, 1937-4763 digital) is published at least six times per year and constitutes the refereed papers of regional conferences sponsored by the Consortium for Computing Sciences in Colleges.

Table of Contents

The Consortium for Computing Sciences in Colleges Board of Directors

Following is a listing of the contact information for the members of the Board of Directors and the Officers of the Consortium for Computing Sciences in Colleges (along with the years of expiration of their terms), as well as members serving CCSC:

Karina Assiter, President (2022), (802)387-7112, karinaassiter@landmark.edu.

Chris Healy, Vice President (2022), chris.healy@furman.edu, Computer Science Department, 3300 Poinsett Highway Greenville, SC 29613.

Baochuan Lu, Publications Chair (2024), (417)328-1676, blu@sbuniv.edu, Southwest Baptist University - Department of Computer and Information Sciences, 1600 University Ave., Bolivar, MO 65613.

Brian Hare, Treasurer (2023), (816)235-2362, hareb@umkc.edu, University of Missouri-Kansas City, School of Computing & Engineering, 450E Flarsheim Hall, 5110 Rockhill Rd., Kansas City MO 64110.

Cathy Bareiss, Membership Secretary (2022), cathy.bareiss@betheluniversity.edu, Department of Mathematical & Engineering Sciences, 1001 Bethel Circle, Mishawaka, IN 46545.

Judy Mullins, Central Plains Representative (2023), Associate Treasurer, (816)390-4386, mullinsj@umkc.edu, UMKC, Retired.

Michael Flinn, Eastern Representative (2023), mflinn@frostburg.edu, Department of Computer Science & Information Technologies, Frostburg State University, 101 Braddock Road, Frostburg, MD 21532.

David R. Naugler, Midsouth Representative(2022), (317) 456-2125, dnaugler@semo.edu, 5293 Green Hills Drive, Brownsburg IN 46112.

Grace Mirsky, Midwest Representative(2023), gmirsky@ben.edu, Mathematical and Computational Sciences, 5700 College Rd. Lisle, IL 60532.

Lawrence D'Antonio, Northeastern Representative (2022), (201)684-7714, ldant@ramapo.edu, Computer Science Department, Ramapo College of New Jersey, Mahwah, NJ 07430.

Shereen Khoja, Northwestern Representative(2024), shereen@pacificu.edu, Computer Science, 2043 College Way, Forest Grove, OR 97116.

Mohamed Lotfy, Rocky Mountain Representative (2022), Information Systems & Technology Department, College of Engineering & Technology, Utah Valley University, Orem, UT 84058.

Tina Johnson, South Central Representative (2024), (940)397-6201, tina.johnson@mwsu.edu, Dept. of Computer Science, Midwestern State University, 3410 Taft Boulevard, Wichita Falls, TX 76308.

Kevin Treu, Southeastern Representative (2024), (864)294-3220, kevin.treu@furman.edu, Furman University, Dept of Computer Science, Greenville, SC 29613.

Bryan Dixon, Southwestern Representative (2023), (530)898-4864, bcdixon@csuchico.edu, Computer Science Department, California State University, Chico, Chico, CA 95929-0410.

Serving the CCSC: These members are serving in positions as indicated:
Bin Peng, Associate Editor, (816)584-6884, bin.peng@park.edu, Department of Computer Science and Information Systems, Park University, 8700 NW River Park Drive, Parkville, MO 64152.
Ed Lindoo, Associate Treasurer & UPE Liaison, (303)964-6385, elindoo@regis.edu, Anderson College of Business and Computing, Regis University, 3333 Regis Boulevard, Denver, CO 80221.
George Dimitoglou, Comptroller, (301)696-3980, dimitoglou@hood.edu, Department of Computer Science & Information Technology, Hood College, 401 Rosemont Ave., Frederick, MD 21701.
Carol Spradling, National Partners Chair, carol.spradling@gmail.com.
Megan Thomas, Membership System Administrator, (209)667-3584, mthomas@cs.csustan.edu, Department of Computer Science, CSU Stanislaus, One University Circle, Turlock, CA 95382.
Deborah Hwang, Webmaster, (812)488-2193, hwang@evansville.edu, Electrical Engr. & Computer Science, University of Evansville, 1800 Lincoln Ave., Evansville, IN 47722.

CCSC National Partners

The Consortium is very happy to have the following as National Partners. If you have the opportunity please thank them for their support of computing in teaching institutions. As National Partners they are invited to participate in our regional conferences. Visit with their representatives there.

Platinum Partner

Google Cloud
GitHub
NSF – National Science Foundation

Silver Partners

zyBooks

Associate Partners

Mercury Learning and Information
Mercy College

Welcome to the 2022 CCSC South Central Conference

The 2022 South Central Steering Committee is very pleased to welcome everyone to our 33rd annual conference online hosted by the Computer Science Department at The University of Texas at Dallas. Our conference chair and host, Shyam Karrah, has provided infrastructure and support for the virtual delivery of our conference. With generous time and effort the Steering Committee has delivered a fine program overcoming the challenges of virtual hosting and production of this year's program.

For our online 2022 conference we have six papers, one workshop, and both student and faculty posters scheduled for the program. This year the Steering Committee chose 6 of 11 papers through a double-blind review process for a paper acceptance rate of 54%. Eighteen colleagues across the region and country served as professional reviewers and we recognize the expertise and guidance they all so thoughtfully contributed to the selection of our 2022 conference program.

The Steering Committee continues to seek colleagues to host the conference in the future and to join our community of computer science educators to enrich our curricula and provide innovative pedagogy for our students. We invite and encourage our fellow members of the South Central region to attend our Monday April 11, 2022 virtual evening business meeting. Fellow educators and colleagues are encouraged to join in our efforts to involve more of our community in the planning and execution of the conference in the future.

We extend a very warm and delightful welcome to our virtual presenters and attendees who continue to promote computer science education and camaraderie to our region. To all members of our 2022 Steering Committee, thank you again for your gracious efforts in delivering our first virtual conference during such challenging times.

Shyam Karrah
The University of Texas at Dallas
Conference Chair and Host

Bingyang Wei
Texas Christian University
Regional Editor Co-Chair

Mustafa Al-Lail
Texas A&M International University
Regional Editor Co-Chair

2022 CCSC South Central Conference Steering Committee

Conference Chair
Shyam Karrah ..... University of Texas at Dallas, TX
Papers Chair
Bingyang Wei ..... Texas Christian University, TX
Mustafa Al-Lail ..... Texas A&M International University, TX
Reviewer Chair
Lasanthi Gamage ..... Webster University, MO
Julie Smith ..... University of North Texas, TX
Panels and Tutorials Chair
Jeffrey Zheng ..... Stephen F. Austin State University, TX
Posters Chair
Michael Scherger ..... Texas Christian University, TX
Moderator Chair
Abena Primo ..... Huston-Tillotson University, TX
Publicity Chair
Eduardo Colmenares-Diaz ..... Midwestern State University, TX
Nifty Assignments Chair
Michael Kart ..... St. Edward's University, TX
At-Large Member
Tim McGuire ..... Texas A&M University, TX

Regional Board — 2022 CCSC South Central Region

National Board Representative
Tina Johnson ..... Midwestern State University, TX
Registrar
Anne Marie Eubanks ..... Stephen F. Austin State University, TX
Treasurer
Bilal Shebaro ..... St. Edward's University, TX
Regional Editor
Bingyang Wei ..... Texas Christian University, TX
Mustafa Al-Lail ..... Texas A&M International University, TX
Webmaster
Abena Primo ..... Huston-Tillotson University, TX

Reviewers — 2022 CCSC South Central Conference

Barbara Anthony ..... Southwestern University, Georgetown, TX
Laura Baker ..... St. Edward's University, Austin, TX
Eduardo Colmenares ..... Midwestern State University, Wichita Falls, TX
Lasanthi Gamage ..... Webster University, Webster Groves, MO
Joshua Gross ..... California State University Monterey Bay, Seaside, CA
David Gurney ..... Southeastern Louisiana University, Hammond, LA
Jayantha Herath ..... St. Cloud State University, St Cloud, MN
Michael Kart ..... St. Edward's University, Austin, TX
Jose Metrolho ..... Instituto Politécnico de Castelo Branco, Castelo Branco, Portugal
Tim McGuire ..... Texas A & M University, College Station, TX
Muhammad Rahman ..... Clayton State University, Morrow, GA
Janet Renwick ..... University of Arkansas Fort Smith, Fort Smith, AR
Michael Scherger ..... Texas Christian University, Fort Worth, TX
Bilal Shebaro ..... St. Edward's University, Austin, TX
Julie Smith ..... University of North Texas, Denton, TX
Bingyang Wei ..... Texas Christian University, Fort Worth, TX
Chadd Williams ..... Pacific University, Forest Grove, OR
Jeffrey Zheng ..... Stephen F. Austin State University, Nacogdoches, TX

Knowing What It Knows and Asking the Right Questions? An Effective Approach to Human-Allied AI*

Keynote

Sriraam Natarajan
Department of Computer Science, University of Texas Dallas

Abstract

Historically, Artificial Intelligence has taken a symbolic route for representing and reasoning about objects at a higher-level or a statistical route for learning complex models from large data. To achieve true AI, it is necessary to make these different paths meet and enable seamless human interaction. First, I briefly will introduce learning from rich, structured, complex and noisy data. Next, I will present the recent progress that allows for more reasonable human interaction where the human input is taken as "advice" and the learning algorithm combines this advice with data. The advice can be in the form of qualitative influences, preferences over labels/actions, privileged information obtained during training or simple precision-recall trade-off. Finally, I will outline our recent work on "closing-the-loop" where information is solicited from humans as needed that allows for seamless interactions with the human expert. While I will discuss these methods primarily in the context of probabilistic and relational learning, I will also present our recent results on reinforcement learning and demonstrate how human input can be effectively used to create appropriate abstractions to guide RL.

Bio

Dr. Sriraam Natarajan is a Professor at the Department of Computer Science at University of Texas Dallas and a RBDSCAII Distinguished Faculty Fellow at IIT Madras. His research interests lie in the field of Artificial Intelligence, with emphasis on Machine Learning, Statistical Relational Learning and AI, Reinforcement Learning, Graphical Models and Biomedical Applications. He is a AAAI senior member and has received the Young Investigator award from US Army Research Office, Amazon Faculty Research Award, Intel Faculty Award, XEROX Faculty Award, Verisk Faculty Award and the IU trustees Teaching Award from Indiana University. He is the AI and society track chair of AAAI 2022, demo chair of IJCAI 2022, program co-chair of SDM 2020 and ACM CoDS-COMAD 2020 conferences. He was the specialty chief editor of Frontiers in ML and AI journal, and is an associate editor of MLJ, JAIR, DAMI and Big Data journals.

Teaching Data Representation Using Real-World Applications*

Daniel Ray, Mark Terwilliger
Computer Science and Information Sciences
University of North Alabama
Florence, AL 35632
{dray4, mterwilliger}@una.edu

Abstract

Students who are majoring in computer-related disciplines should have a reasonable understanding of data representation. Concepts like encoding, range and precision, overflow, round-off error, and data compression are all vitally important. Unfortunately, students often struggle to recognize the practicality of these concepts. To address this concern, we created a 6-module lesson plan that uses real-world examples to help students connect with this material. When incorporated into our Computer Science I (CS1) course, we found measurable improvements in both student perceptions and associated computational skills.

1 Introduction

Number systems are one of the most fundamental concepts in computer science and engineering education[8]. As computer science educators, we strongly believe that our students should understand number systems, as well as computing related concepts like range and precision, overflow, round-off errors, encoding schemes, and data compression. In our experience, students have shown resistance when these ideas are presented, and struggle to find their practical relevance. To address this, we created a 6-module, 1-week lesson based on real-world applications for inclusion in our CS1 course. Additionally, we created tools to measure the lesson's impact on student outcomes.

When conducting this preliminary study, our hypothesis was that lessons built around real-world examples would help improve student attitudes, understanding, and skills related to number systems and data representations in a relatively short window of time. We believe our results support this hypothesis.

2 Previous Work

We relied on several important works in developing our lesson plan. One classic paper relies on the theory of multiple intelligences to suggest alternatives to using classic linguistic or mathematical approaches when teaching data representation[2]. Other works explore alternative methods and curricula for teaching number systems and data representation at the K-12 level[7] and using interactive software to present this content[1]. Other works guiding our research included an effort to teach data science to non-CS students[4] and a review of efforts to use problem-based learning as a paradigm for an entire CS program[6]. Our work aligns with and expands upon these findings.

3 Methodology

Our study uses a cognitive constructivist epistemology which argues that deep learning occurs when students can contextualize new content by relating it to practical experiences[3]. To accomplish this, we developed a two-day lesson in a CS1 class introducing number systems and data representation. It consisted of 6 modules focusing on relatable real-world examples highlighting skills related to these areas. College pre-algebra is a prerequisite and discrete structures a corequisite for the course. Students received no additional priming instructional content before the study. We presented the first three modules on day one and the final three on day two. To measure the learning outcomes, we also gave students a paper-based survey both before and after the lesson. We discuss the modules in more detail in the next section.

For our study group, we used two sections of the CS1 course at our institution during the 2021 Fall semester. The student pool for this course is a representative cross-section of students in computing as well as mathematics and technology-related fields in various majors. While there were 48 students enrolled in these two sections, only 33 students were present for both lessons and completed both the pre-survey and post-survey. Therefore, we only included these 33 students in our results.

The survey consisted of ten questions and was designed to be completed in approximately 15 minutes. The questions included three direct measures of relevant skills questions, two Likert-type scale measures of student attitudes

towards the importance of the material, and five open response questions that tested students' mental models toward the material and its significance.

4 The Modules

An overview of the six modules is shown in Table 1. In each module, we present some real-world applications, cover data representation concepts, and have students take part in hands-on activities.

Table 1: The Modules

Mod.	Title	Concept	Activities
A	Birthday Cakes	Number Systems, Binary Number System	1. Convert Decimal to Binary 2. Convert Binary to Decimal
B	Parachute	Binary Encodings	1. Use Binary to Decode a Message
C	Morse Code	Data Compression Encodings: (a) Keyword, (b) Run-length, (c) Huffman Compression Ratio	1. Encode/Decode using: (a) Keyword, (b) Run-length, (c) Huffman 2. Compute Compression Ratio
D	What's in a Color?	Using Binary to Store: (a) Black and White Image, (b) Color Image Hexadecimal Number System	1. Convert Decimal to Binary to Recover an Image 2. Convert Decimal to Hex to Create Web Page Color Scheme
E	What Time is it?	Representing Time, The Overflow Problem	1. Convert Decimal to Binary / Hex 2. Other Numeric Conversions
F	Deadly Round-off Error	Representing Floating Point Round-Off Errors	1. Convert Fractions to Binary 2. Compute Round-Off Errors

4.1 Module A – Birthday Cakes

The first module introduced number systems starting with dozenal (base 12), vigesimal (base 20), and sexagesimal (base 60) and discussed why these were used in antiquity. As we transition to binary (base 2), we give each student a

whole number "Bit Flipper," which is one of two tools (see Figure 1) we developed based on the code.org "Flippy Do"[5]. Students used their Bit Flipper to practice conversions between binary and decimal. Finally, students discovered how to conserve candles on birthday cakes using binary counting.

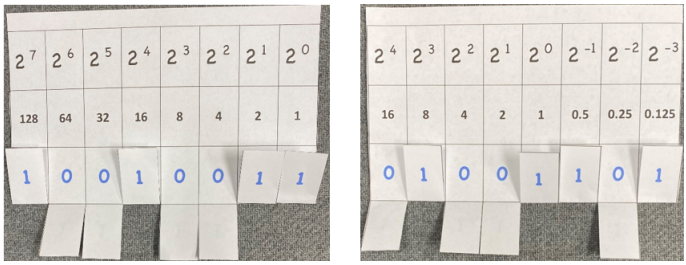

Figure 1: Bit Flippers version 1 (whole numbers) and version 2 (fractional)

4.2 Module B – Parachutes

We reinforced the concept of different number systems and introduced binary encodings of text in the second module. To answer, 'How can we represent human languages?' we examined a secret message scheme that was used on the Mars 2020 lander parachute (see Figure 2). After breaking down the encoding scheme used and discovering the secret message ("Dare Mighty Things"), students were given a parachute containing a different message to decode on their own.

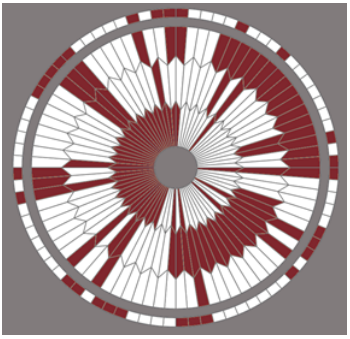

Figure 2: A secret message encoded in the Mars 2020 Lander parachute.

4.3 Module C – Morse Code

In Morse code, the letters E and T, which are used frequently, are represented by a single pulse, while less frequently used letters like Q and X are represented using four pulses each. The class discussed the advantages of a variable-sized encoding scheme such as this one. In the module, students also talked about reasons why we would want to compress data. Three compression techniques were presented, including keyword encoding, run-length encoding, and Huffman encoding. Students used these techniques to encode and decode strings, as well as compute compression ratios.

4.4 Module D – What's in a Color?

An introduction to the hexadecimal number system showed students that binary numbers can be used to represent not only characters, but also multimedia data such as colors and images. We started by looking at a black-and-white image, using one bit per pixel. By increasing the number of bits per pixel, our color palette increased. Students put this knowledge to work to create a simple web page color scheme. To do this, students had to convert RGB (Red Green Blue) values to their 6-character hex-code equivalents.

4.5 Module E – What Time is it?

A discussion of the famous Y2K event helped students learn the concept of overflow and the importance of choosing an appropriate date representation scheme. This led into a related discussion of Y2K38, a similar event on the horizon concerning UNIX time. Other related conversations involved the rolling over of an older style mechanical car odometer and an overflow glitch in the Pac Man arcade game. A hands-on activity involved students picking a date and time, finding the equivalent UNIX Timestamp, and then computing how long before an overflow event occurs.

4.6 Module F – Deadly Round-off Error

In our final module, we introduced a binary fractional representation and had students use Bit Flipper version 2 (see Figure 1) to represent fractional numbers. Students soon discovered the concept of round-off error. A discussion of a fatal 1991 failure of the Patriot missile defense system, which occurred because of a fixed-point round-off error, illustrated the potentially serious consequences of poor data storage design decisions.

5 Results

In this section we analyze our results by first arguing that a significant improvement can be seen in related skill tests. Next, we argue that student attitudes concerning the importance of the material deepened and that their perception of the material matured.

5.1 Quantitative Data – Skills Inventory

One hypothesis that we wanted to test with this study is that lessons in data-representation using real-world examples would help students learn relevant skills in a short amount of time. To test this, we asked students to exhibit those skills in a series of questions on the survey. If our hypothesis is true, we would expect to see a marked improvement in those skill sets in the post-survey over the pre-survey.

Question 5 asked students to convert the binary value 101101 to its decimal form, Question 6 asked students to convert the decimal value 75 to binary, and Question 7 asked students to convert the decimal number 75 to hexadecimal. This sort of question ensures that the original representation is understood by the student, and also that the student possesses sufficient knowledge of the second number system to complete the conversion. Pre-survey results found student knowledge widely lacking.

We classified results as successful, unsuccessful, and "total beginner," where unsuccessful responses at least exhibited some numerical sense, while "total beginner" results reflected non-numeric answers. For converting binary to decimal, only 13 of 33 participants, or 39%, were successful, while 36% were classified as total beginners. Only 12 participants (36%) successfully converted from decimal to binary with fully 51% falling in the "total beginner" category, and only 2 participants (6%) were able to convert from decimal to hexadecimal with 75% in the "total beginner" category.

The post-survey results showed dramatic improvement. Binary to decimal was completed with a success rate of 93% with no students in the total beginner category. Decimal to binary was completed with 100% success rate, and decimal to hexadecimal was completed with a 70% success rate. These improvements constitute solid measurable evidence of significant improvement in skills related to the material learned over a very short interval.

5.2 Qualitative Data – Student Attitudes

The Likert-type questions asked on the pre/post-survey were geared toward testing our second hypothesis that lessons built around real-world examples would improve student attitudes toward these topics.

Question 1 on the survey asks, "On a scale of 1 to 5, how important do you think knowledge about number systems is for the average person?" Question 2 asks the same question but for "a career in a computer-related field."

The pre-survey results showed an average response of 3.59 for perceived importance to the average person and a 4.41 average response for perceived importance to a career in computer-related fields. One possible explanation for the relatively low score for the "average person" is that students in the study often seemed to have a faulty idea of what was meant by "number systems". Still, a positive shift in this number, especially from pre- to post-survey, would be strong evidence for improvement in learning and understanding. In addition, while the perceived importance of number systems to a career in computer-related fields is high in relation to the average person, we would expect it to be rated higher still overall for such a fundamental part of computer science.

The data from the post-survey exhibit a positive shift. Students answered Question 1 at an average of 3.91, an 0.33-point increase (plus 6.6%), and Question 2 at an average of 4.88, a 0.47-point increase (plus 9.4%). This indicates that an exposure to real world examples where number systems and data representation play a key role increased knowledge and matured student perceptions.

5.3 Qualitative Data – Student Perceptions

The remaining five questions were open-response questions that aimed to qualitatively capture any changes in student perception and understanding regarding these topics.

Question 3 asked: "Name a couple of instances that you can think of where you have come across various number systems outside of schoolwork." Responses to this question in the pre-survey support the idea that there was confusion amongst respondents as to what was meant by "number systems". Responses on the post-survey showed less overall confusion.

For instance, one student who responded, "I'm not really sure" on the pre-survey answered, "working on html, cybersecurity competitions" on the post-survey, showing that they were able to link their personal experiences to the topic as a result of the lesson.

Three other students on the pre-survey answered, "Counting money", "Cashier – making change", and "Banking or figuring discounts". This was a common misconception that the only number system outside of schoolwork deals with amounts (presumably in base 10) and often dealing with financials.

These same students answered differently on the post-survey. They answered, "Unix timestamp, birthday dates" (the reference to birthday dates referring apparently to both the elementary counting example of binary birthday candles as well as the lesson on calculating your birth date and time using

the Unix epoch), "computer data, phone numbers", and "coding or making a website", respectively.

Question 4 asked: "Name three things you think the following value might represent: 46166B" Responses varied widely on an open question such as this one. Overall, students showed improvement from pre-survey to post-survey. We highlight a few student responses that typify this improvement below.

On the pre-survey, students answered things like: "4 6 4 6 6 11, n/a, n/a", "I don't know, n/a, n/a", and "46 billion, n/a, n/a". On the post-survey those same students answered: "hexadecimal so could be a color value, date, binary value", "A color, a number in hex, 0100 011 0001 0110 0110 12", and "A number that has been converted to hex, 0100 0110 0001 0110 0110 1011, 4616611", respectively.

Clearly, these responses show a level of reasoning about number systems that is still developing but also showed a noticeable evolution.

Question 8 asked: "What specifically about a number system does the "base" of that number system determine?" Non-responses, "N/A", "I don't know", "IDK" or something similar dominated the pre-survey (39%). Of the rest, many were answers such as, "The "key"", "the start", "The value?", "the height?? of the number system", etc., that showed students who were either not at all or only vaguely familiar with the term and likely the concept of different number systems. A total of 10 students (another 30%) gave such "beginner" answers.

Post-survey responses were expectantly varied but again showed a marked improvement. Seventeen students (51%) correctly identified what was the base of a number system. Another four students answered the question by referencing number systems of specific bases. Finally, six more students answered by saying that the base represents the number of "digits" in the system or the "size" of the system; confused and imprecise to be sure, but a confusion of terminology only and still an improvement. Even among the remaining six students, the answers, while wrong, revealed the respondents to be emergent learners.

Question 9 asked: "What are the advantages and disadvantages to using a number system with a larger base compared to a smaller base?" An advanced level of understanding will focus on the tradeoff between a larger base with fewer symbols required to store large numbers but more overhead in remembering larger sets of unique symbols and a smaller base with fewer symbols to represent but increasingly larger numbers to store data.

In the pre-survey responses, we see little to suggest sophisticated understanding with 18 responses (55%) being either fully (both advantages and disadvantages) or partially (one or the other) left as non-answers.

The post-survey results offer a mixed bag with most students still exhibiting confusion. The number of non-answers dropped from 18 to 2. However, all

students exhibited maturing mental models.

Finally, Question 10 asked: "What information do you have to consider to decide what the underlined value in the following number means? 723048276" On the pre-survey, 19 students (58%) either left this question completely or partially blank. Among those counted in this group, the partially blank responses exhibited novice level understanding of the question's implications only.

In contrast, among post-survey responses, only three students (9%) fell into this category. A wide range of answers was present on the post-survey, which frustrates generalization about student understanding. However, we do note that a total of 21 students (63%) in their responses mentioned either the base of the number (10 total), the position of the number (3 total), or both (8 total). This is a marked improvement over the pre-survey results, where only 13 students (39%) fell into the same category with only 3 respondents mentioning the base of the number system, 5 mentioning the place value, and 5 mentioning both.

6 Conclusion

Our hypothesis when conducting this preliminary study was that lessons built around real-world examples would help students to both grasp the topics and skills related to data representation as well as to improve student attitudes toward and deepen the complexity of the understanding about these topics. These results could also be realized in a short window of time. Our results show improvement in all three areas: knowledge retention, student perceptions about covered concepts, and student mental models. These improvements were significant enough to warrant further study and expansion of our methodology.

References

[1] Ruedi Arnold, Marc Langheinrich, and Werner Hartmann. InfoTraffic: teaching important concepts of computer science and math through real-world examples. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 105–109, 2007.

[2] Katrin Becker. A multiple intelligences approach to teaching number systems. *Consortium for Computing Science in Colleges Northwest Conference*, 2003.

[3] Mordechai Ben-Ari. Constructivism in computer science education. *Acm sigcse bulletin*, 30(1):257–261, 1998.

[4] Robert J Brunner and Edward J Kim. Teaching data science. *Procedia Computer Science*, 80:1947–1956, 2016.

[5] Code.org. Binary numbers. `https://curriculum.code.org/csp-20/unit1/4/`, 2017. Accessed: 2021-12-09.

[6] Steve Cooper and Steve Cunningham. Teaching computer science in context. *Acm Inroads*, 1(1):5–8, 2010.

[7] Yvon Feaster, Farha Ali, and Jason O Hallstrom. Serious toys: teaching the binary number system. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education*, pages 262–267, 2012.

[8] Manuela Panoiu, Anca Iordan, Caius Panoiu, and Loredana Ghiorghioni. Educational software for teaching the basics of computer science. *WSEAS Transactions on Advances in Engineering Education*, 12(6):238–243, 2009.

Motivating the Study of Discrete Structures with Experiments*

Jose Cordova
Department of Computer Science and Computer Information Systems
University of Louisiana at Monroe
Monroe, LA 71203
cordova@ulm.edu

Abstract

In this paper, we discuss the use of experiments to motivate and reinforce the study of selected topics in the field of discrete mathematics. Experiments are used to verify hypotheses based on students' solutions to problems of moderate complexity. Examples are presented from the areas of summations, probability, recursive sequences, and relations. For each experiment, a procedure is outlined to test the experimental hypothesis. The paper concludes with suggestions for classroom use and possible avenues for future research.

1 Introduction

For the past several years, there has been a renewed emphasis on enhancing the role of discrete mathematics in undergraduate Computer Science curricula [9]. It is widely recognized that discrete mathematical structures are vital to the type of computational thinking essential for effective abstraction and problem solving in computer science and software engineering ([4], [7]). In this light, significant effort has been devoted to the development of hands-on activities and demonstrations for use in a classroom setting [3].

Increasingly, computer science educators face the difficulty of impressing upon students the notion that discrete structures are not only important but

indeed very relevant to their chosen profession. As discussed by Gries [6],"... students ask 'Why do we have to learn all this?' as they study sequences, mathematical induction, counting, probability, etc., relatively early in their college years. Unfortunately, the most convincing answer comes during the subsequent semesters, and it comes in many iterations." The question is then, not only how to motivate the study of discrete structures concepts, but what the appropriate timing is for the introduction of these topics in an undergraduate computer science curriculum [9].

We suggest that part of the answer may be found in the field of experimental computer science, which focuses on the role of experimentation as "necessary and beneficial both as a complement to theory and as an element in constructing systems" [5]. Even though empirical studies can both confirm and complement the results of theoretical analysis [8], this practice has been traditionally underutilized in computer science education, unlike what has been commonplace in the natural sciences [1].

In this paper, we discuss the use of experiments to motivate and complement the study of discrete mathematical structures in computer science. The paper describes four experiments and concludes with observations regarding their uses in a classroom setting.

2 Use of Experiments in Teaching Discrete Structures

In the traditional scientific method, experiments are conducted to test hypotheses in a particular subject of study. As such, an experiment traditionally includes a statement of the problem, a formulation of the hypothesis being tested, a description of the procedure or methodology used for testing the hypothesis, observation and collection of data, and the establishment of conclusions based on analysis of the data. As pointed out before, experiments are useful in confirming theoretical results. In addition, empirical studies often point scientists in the direction of new theories.

Topics in a discrete structures course can be motivated by the presentation of problems whose solution requires the correct application of mathematical concepts. When working exercises and arriving at potential solutions theoretically, students often wonder whether the potential solution is correct. Verification of the theoretical result is usually limited to a comparison with model answers provided by the instructor, the textbook, or their peers. In the worst case, verification fails to take place, potentially leaving students uncertain regarding the correctness of their solution.

We suggest that experiments can be a valuable tool in motivating the study of discrete structures and in helping students confirm - or refute - the correctness of their theoretical results. In the following examples, typical problems

related to discrete structures topics are presented, as well as sample experiments to validate the theoretical solutions. Completion of the experiment usually - though not always - involves writing a program or program segment, simple enough to be within the reach of a student who has completed an introductory programming course. Unlike approaches that use specialized languages such as Alloy [2], the experiments presented here can be implemented using any general-purpose programming language.

3 Experiment 1: Sequences and Summations

The objective of this experiment is to verify the correctness of a theoretical solution to the problem of simplifying an expression involving summations. For instance, students are asked to simplify an expression such as

$$\sum_{j=1}^{200} \frac{3j}{2} \quad + \quad 3\sum_{k=0}^{199} \frac{k+3}{2} \tag{1}$$

The correct solution requires application of the distributive property of summations, although before the summations can be combined, one needs to change one of the index variables in order to obtain a match of the lower and upper limits. The variable change (for instance, j = k + 1) needs to be propagated in the corresponding summation, resulting in an equivalent expression such as

$$\sum_{k=0}^{199} (3k+6) \tag{2}$$

In the author's experience, students find it difficult to simplify this type of expression correctly. Therefore, experimental verification is not only appropriate, but a potentially useful tool in discovering errors. Once the student arrives at a potential solution (such as expression (2)), an experimental hypothesis can be formulated stating that expression (2) is equivalent to expression (1). To test the hypothesis, students can be instructed to:

- write a program segment that implements the operations in expression (1), including one loop for each summation and a statement to combine the results
- write a program segment that implements the operations in expression (2), which uses a single loop construct
- observe and compare the results from both program segments

Students can then form a conclusion based on the results. If correct, the student confirms that the alternative expression is not only equivalent to the original expression, but algorithmically simpler and computationally less expensive. This conclusion can be a powerful motivating factor. A negative conclusion means that either the alternative expression (2) or the software implementation of the experiment is flawed. One possible variation of the experiment is for the instructor to derive expression (2) and ask the students to find and test another alternative expression by changing a different variable. Another variation is to start with a program segment containing two loops, ask the students to translate the code into an equivalent expression involving summations, simplify the expression, and test its correctness by implementing the corresponding program segment.

4 Experiment 2: Counting and Probability

The study of basic probability lends itself to a variety of experimental possibilities. This example is a variation of a problem described by Winkler [10]. In this example, given the fact that the probability of winning a bet at the roulette table is 1/38, that placing a bet costs \$1, and that a win earns the gambler \$36, the students are asked to derive the probability that, in a run of 105 bets, the gambler will win exactly 0, 1, or 2 times (and therefore lose money, since the gambler must win at least three times in order to come out ahead). A general description of the solution, which is not shown here due to space limitations, is presented by Winkler [11]. As is the case with the first example, since calculating the correct probability may be difficult for some students, experimental verification can be helpful.

Given the difficulty students may have in calculating the probability that the number of wins is less than three, it is probably best to approach the problem in steps. First, ask students to find the probability that the gambler will not win at all in 105 attempts. Students should calculate the result as $(37/38)^{105}$. Once this is accomplished, the experimental hypothesis to be verified is that this theoretical result (approximately 0.06) is, in fact, the probability of not winning at all. The procedure used to test the hypothesis can be described to student as follows:

- pick a positive integer less than or equal to 38 as your bet
- use a random number generator to sample 105 integers in the range 1 to 38, including code that counts the number of times (out of 105) that your bet is selected
- complete the program by including a loop that repeats the above steps n times, for n = 10, 100, 1000, ..., and computes the number of trials

in which the gambler won no bets (this value, when divided by n, can be used to approximate the desired empirical probability)

After observing the results and summarizing the data, students should be able to arrive at a conclusion regarding their calculated probability. If correct, the empirical probability obtained by the program should converge towards the theoretical probability as the value of n increases. The fact that the value of n must be sufficiently large in order to obtain reliable results is a valuable lesson for students.

A similar process can be used to compute and verify the probability of winning exactly once and the probability of winning exactly twice in 105 tries. Finally, the overall probability of winning less than three times, which can be computed by adding the three partial results, can be verified by making a small modification to the program. The experiment provides an opportunity to discuss issues regarding the law of large numbers, random sampling, and the use of simulation for estimating results when theoretical results are difficult to compute.

5 Experiment 3: Directed Graphs and Binary Relations

The objective of this experiment is to verify the student's ability to produce the transitive closure of a relation by comparing it to the result obtained through the application of Warshall's algorithm. A secondary objective, from a pedagogical point of view, is to reinforce the connection between the concepts of binary relations, directed graphs, and matrices.

Students are provided with the elements of a finite set S and with the ordered pairs that make up a relation R on S. They are asked to draw the corresponding directed graph and to use the graph to list the ordered pairs in the transitive closure of R, as suggested in various discrete structures textbooks. Typical examples used include a set of courses in a college curriculum and the ordered pairs that result from applying the "is prerequisite of" relation, or a set of users who are members of a social network and the ordered pairs that result from applying the "is a friend of" relation. Unless S and R are trivially small, the process of generating the transitive closure of R by hand can be time consuming and prone to error. An experiment can help students validate the results by testing the hypothesis that the set of ordered pairs generated from the directed graph is in fact the transitive closure of R.

The procedure used to test the hypothesis consists of the following steps:

- use the original set of ordered pairs in R to construct the adjacency matrix for R
- use Warshall's algorithm to produce the corresponding reachability matrix

- compare the set of ordered pairs in the transitive closure with the ones in the reachability matrix

The last step of the process allows the student to confirm or reject the stated hypothesis. Initially, students can be asked to produce the reachability matrix by doing a hand trace of the algorithm. Alternatively, students can implement the algorithm in an actual program.

6 Experiment 4: Recursively Defined Sequences

The objective of this experiment is to verify the student's ability to produce a closed form for a recursively defined sequence, such as

$$A_n = 2A_{n-1} + 3 \quad for \quad n > 1 \tag{3}$$
$$A_1 = 5$$

The result, which can be obtained either by iteration or by the application of the solution formula for linear first-order recurrence relations, can be expressed as

$$A_n = 8(2^{n-1}) + 3 \quad for \quad n \geq 1 \tag{4}$$

Given the difficulty many students experience in deriving expressions such as (4), verification is an important step. Though the first principle of mathematical induction provides a formal and elegant way to verify that (3) and (4) produce the same sequence, it has been our experience that many students have just as much trouble in constructing an argument based on mathematical induction as they would in solving the original recurrence. Empirical evidence –though itself not a proof- can help verify the hypothesis that the sequence produced by the expression in (4) is equivalent to that produced by the recursive definition in (3). The experiment can be completed as follows:

- implement a recursive subprogram to model the sequence defined in (3)
- implement a loop-based algorithm that repeatedly invokes the recursive subprogram and evaluates the expression in (4) for n = 1 to a sufficiently large upper limit
- compare the values returned by the recursive method with the results of evaluating the expression in (4) for all values of n

Again, it is important to emphasize to students that empirical data is to be considered simply as evidence in support of the stated hypothesis, not a replacement for a rigorous proof. In this regard, the experiments described in this paper play a similar role as experiments designed to test hypotheses in the natural sciences.

7 Summary and Observations

The use of experimentation can be effective in motivating the study of discrete structures and in helping students verify the correctness of their solutions. Computer science students who are usually more interested in programming than in mathematics have the opportunity to apply their programming skills in the completion of the experiments. As an interesting side effect, in cases when the theoretical results differ from the empirical results, students are forced to ponder and investigate as scientists often do: is the theoretical result incorrect? or are there flaws in the implementation of the experiment?

The examples provided are drawn from the topics of probability, summations, recursive sequences, and relations. Various others are possible. The type of experiments described in this paper can be assigned in a classroom setting, as homework, or as a combination. The experiments are well suited for use in small groups in an effort to foster collaborative learning. Though this is not always the case, the experiments usually involve programming tasks, with the instructor providing starter code if necessary. One must be careful to ensure that the program be simply considered as a tool for the validation of the theoretical results. Accordingly, the programming activity must be simple enough so as not to distract from the purpose of the experiment.

Further research involves an investigation of the effectiveness of the use of experiments in fulfilling the learning outcomes of a discrete structures course. This research would include a series of pre- and post-experiment quizzes designed to measure whether there is a significant change in students' performance as a result of experimentation in a particular topic.

References

[1] Victor R Basili and Marvin V Zelkowitz. Empirical studies to build a science of computer science. *Communications of the ACM*, 50(11):33–37, 2007.

[2] Laura E Brown, Adam Feltz, and Charles Wallace. Lab exercises for a discrete structures course: Exploring logic and relational algebra with alloy. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, pages 135–140, 2018.

[3] Lijuan Cao and Audrey Rorrer. An active and collaborative approach to teaching discrete structures. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*, pages 822–827, 2018.

[4] Michael E Cotterell, Delaram Yazdansepas, and Bradley J Barnes. Active learning in cs2 and discrete mathematics. In *Proceedings of the 51st ACM*

Technical Symposium on Computer Science Education, pages 1318–1318, 2020.

[5] Dror Feitelson. Experimental computer science. *Communications of the ACM*, 50(11):24–26, 2007.

[6] David Gries. Discrete mathematics/structures: How do we deal with the late appreciation problem? *Journal of Computing Sciences in Colleges*, 24(6):110–112, 2009.

[7] Peter B Henderson, Thomas J Cortina, and Jeannette M Wing. Computational thinking. In *Proceedings of the 38th SIGCSE technical symposium on Computer science education*, pages 195–196, 2007.

[8] Catherine C McGeoch. Experimental algorithmics. *Communications of the ACM*, 50(11):27–31, 2007.

[9] Norman Neff. Problem-directed discrete structures course. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 148–151, 2010.

[10] Peter Winkler. Puzzled: Probability and intuition. *Communications of the ACM*, 52(8):104–104, 2009.

[11] Peter Winkler. Puzzled: Solutions and sources. *Communications of the ACM*, 52(9):110–110, 2009.

Recruiting Technology Majors through High-Impact Educational Practices (HIPs)*

Alana Platt, Anna Land, and Andrew Ciganek
Information Technology and Supply Chain Management
University of Wisconsin-Whitewater
Whitewater, WI 53190
{platta, landa, ciganeka}@uww.edu

Abstract

This study proposes integrating a High-Impact Educational Practice (HIP) in an introductory technology course to improve attitudes toward and better attract students to computing majors. Faculty at the University of Wisconsin - Whitewater worked with a community partner to develop technology case studies and administer them to students in an introductory technology course. Students who participated in the case studies exhibited greater gains compared to a control group in their self-assessed familiarity with computing concepts and more positive attitudes towards technology as a viable major.

1 Introduction

The faculty of the University of Wisconsin - Whitewater have been exploring ways to attract undergraduate students to computing majors earlier in their college careers. Based on anecdotal student feedback and internal surveys, over eighty percent of students enrolled in an information technology major started college as a different major. Many students were unfamiliar with the material covered in the major or what an information computing professional does. Further, the faculty sought to identify ways to better support underserved students

in their courses. To this end, the faculty proposed modifying an introductory information technology course to include High-Impact Educational Practices (HIPs), which are a series of high student engagement educational activities which are positively associated with a number of desirable curricular outcomes [7]. The faculty posited that by strengthening the educational experience in introductory courses, students would leave with a more positive opinion of both the area of study and their capacity to succeed in a technology-related career.

This study proposes integrating a community-based learning experience in a lower division course to positively affect student attitudes towards computing majors. A discussion of pertinent studies specific to computing curricular experiences is addressed next in the literature review section. The subsequent methodology section describes the technology-oriented case studies piloted, refined, and then administered over the duration of two semesters with the participation of a community partner as well as the pre- and post-treatment surveys conducted to measure student attitudes. The study results are then presented followed by a discussion and avenues for future work.

2 Literature Review

High-impact practices (HIPs) are a series of high student engagement educational activities which are positively associated with a number of desirable outcomes [7]. The most commonly reported positive outcomes include student persistence, improved academic performance and faculty and peer interaction, as well as increases in intellectual and practical skills [4]. Further, HIPs typically have compensatory effects or accelerated growth for underserved students (i.e., underrepresented minorities, gender, financial aid status, first generation status, etc.) [7]. Integrating captivating classroom experiences into introductory STEM courses offer promising results i.e. [2, 5, 6]. Other prior works provide evidence on the benefits of communication and engagement between students and community partners e.g. [3, 8, 12]. Elements of the assignments include oral communication, critical thinking, working in teams, and real-world application of skills and knowledge, all of which have been recently rated as the most important college learning outcomes by executives and hiring managers [1]. Prior research has investigated the impact of HIPs on aspects of a technology curriculum. One prior work found that HIPs embedded in an undergraduate curriculum led to strong soft skills in the workforce [10]. The authors in [9] studied an expansion of HIPs in an undergraduate cybersecurity degree program and found evidence that they benefited students. Another prior work found that the incorporation of HIPs, specifically learning communities into the first-year computer science experience, led to better student performance and retention [11]. However, little research has examined the use of HIPs in

attracting students to a computing major.

3 Methodology

Survey results from the University of Wisconsin - Whitewater (N=1100) reveal that relatively few undergraduate students participate in university-sponsored HIPs (e.g., learning community, undergraduate research, global experiences, etc.). Many college departments at the University of Wisconsin - Whitewater do offer HIPs within their programs and curricular offerings like collaborative assignments and projects, service learning, community-based learning, and capstone courses. However, most of these HIPs are exclusive to upper-division courses and the faculty hypothesized that incorporating HIPs in lower-division courses would stimulate interest in computing courses at an earlier stage of a student's academic career. The faculty at the University of Wisconsin - Whitewater partnered with a community partner, Liquid Freight, to develop case studies for an introduction to information systems course. Liquid Freight initially reached out to faculty to solicit talent for open positions, but an opportunity for enhanced exposure through semester-long case studies and live partner engagement was subsequently discussed as a rewarding opportunity for the company, students, and faculty. Liquid Freight was founded in the 1930s and is a medium-sized, family-owned logistics firm headquartered in Wisconsin, specializing in food and beverage transportation.

Faculty members worked with a senior executive at Liquid Freight by identifying information technology challenges in their organization that could be developed into case studies. After a series of interviews, three topics were selected that were both of urgent importance to Liquid Freight and aligned with an expertise area of each participating faculty member. Three faculty members separately drafted a case study and shared it with the senior executive and one of the executive's direct reports for multiple rounds of feedback. Once the senior executive and the direct report agreed that a case study accurately captured the problem, the case study was sent to the other two faculty members for a final review.

Each of the three case studies had distinctive learning outcomes appropriate for sophomore-level students in an introductory information technology course. The first case study addressed issues related to a corporate enterprise system not being user-friendly. Liquid Freight had hired a third-party vendor to replace their legacy enterprise system with a custom software as a service (SaaS) solution. Students developed an implementation plan along with a strategy for launching this SaaS platform for partner companies. The second case study endeavored a broader understanding of the role of software and the Internet of Things (IoT) in managing transparency and risks in multi-tier supply

chains. Students learned about operation on tight margins, where investments in emerging technologies are often relegated as excessive and unnecessary unless done in response to a significant negative event or are mandated by a relatively powerful partner. This is a challenge not unique to Liquid Freight nor the transportation industry. The third case study explored the complexity of launching a mobile component of the SaaS solution, the influence of user demographics and user environment on software design, and ethical considerations in software design. Usability implications specific to the trucking industry were explored. In each of the case study assignments, students worked in groups to analyze the case study and propose solutions to questions raised by Liquid Freight. Each case study included a description of the problem followed by analysis questions. These questions addressed both technological and business aspects of the problem and were appropriate in technical content for an introductory information technology course.

A pilot study was conducted with one section of an introduction to information systems course to evaluate the classroom appropriateness of the newly created curricular materials and improve the study design prior to a broader implementation. The senior executive from Liquid Freight attended a lecture of the class to explain their business, their challenges, and to introduce the case studies at a high level. Students were given the opportunity to ask questions for clarification. The session was recorded for use in future semesters. Over the course of the semester, the case studies were assigned as small group problems. At the end of the semester, students were given the opportunity to provide feedback on the case study via a survey and directly to the instructor.

Overall feedback from students was positive. Students generally found working on the case studies to be a positive experience and helped them to better understand the role of information technology in industry. Others stated that having the guest speaker from industry was enjoyable. However, some students seemed unsettled by how "disorganized" Liquid Freight is and how Liquid Freight "doesn't know what they want." The faculty concluded that student expectations for how a business operates does not match reality and that additional background information on innovation would be appropriate so students understand that uncertainty is common in business and not "bad." Additionally, the second case study was confusing to several students. After reviewing the feedback, the faculty concluded that the second case study was too open-ended for the students and decided to omit it from the study. The following semester, the same faculty member taught two sections of the introduction to information systems class: one section received the case study treatment (Class Test; N=59) and one section was the control group (Class Control; N=58). To control for the impact on learning for an additional two assignments, the control group was given a group paper assignment on technologies covered in the case

study addressing logistics and mobile applications. Students in both classes were provided with a pre- and post-survey to assess (1) their self-assessment of information technology knowledge and competency, and (2) attitudes towards their professional field. The survey items were adapted from the Student Assessment of their Learning Gains (SALG) instrument, which focuses exclusively on students' self-reporting of their learning and the degree to which specific aspects of the course have contributed to that learning [11]. Students were offered extra credit if they submitted proof that they had completed the survey. Responses were anonymous and students were given a unique ID that allowed faculty to match pre- and post-survey responses. Responses for students who participated in only one survey were eliminated, resulting in a total of 32 usable responses (54.25% response rate) from the test class and 20 usable responses (39.33% response rate) from the control class.

4 Results

Students were asked to self-assess their familiarity with the seven major topics in the introduction to information systems course. Students were prompted to "Rate your familiarity with the following topics" followed by a set of Likert scale (1= not at all; 5 = a great deal) corresponding to each topic. The results for the technology self-assessment are shown in Table 1. Students in both groups saw statistically significant increases in their assessment scores for all areas. However, students in class test saw greater improvements than those in class control for all areas except for Cybercrime and Information System Security.

The survey also measured attitudes towards the profession using the same Likert scale described above. The specific statements along with results are shown below in Table 2. Class test saw greater statistically significant improvements in: enthusiasm for the subject; feeling they should share their skills through community outreach; the ability to communicate business concepts to others; confidence that they can succeed in the subject area; and likelihood to seek input from outside the discipline in the decision-making process. Class control saw a greater interest in taking other courses in the subject, although it was not statistically significant.

Interestingly, both classes saw decreases in: willingness to seek help from others on academic problems; consideration of end users on project design and communication; incorporating diverse perspectives from those outside the discipline; and sharing expertise through volunteerism (the latter was statistically significant).

Table 1: Responses for Technology self-assessment

Test-Pre	Test-Post	Diff.	Control-Pre	Control-Post	Diff.
Information Systems in Organizations					
3.21(0.65)	4.18(0.98)	0.97**	3.57(0.87)	4.42(0.81)	0.86**
Business Intelligence and Analytics					
2.91(0.91)	4.09(0.98)	1.18**	3.33(0.97)	4.43(0.75)	1.10**
Database Systems and Big Data					
2.73(0.67)	3.94(1.03)	1.21**	3.00(0.84)	4.10(0.89)	1.10**
Hardware and Mobile Devices					
3.61(0.93)	4.45(1.12)	0.85**	3.62(1.02)	4.38 (0.86)	0.76**
Software and Mobile Applications					
3.42(1.06)	4.53(1.05)	1.11**	3.67(0.91)	4.52(0.87)	0.86**
Electronic and Mobile Commerce					
3.21(1.02)	4.31(0.90)	1.10**	3.38(1.02)	4.48(0.87)	1.09**
Cybercrime and Information System Security					
2.75(0.76)	4.03(1.05)	1.28**	2.76(0.94)	4.10(0.89)	1.33**
** significant to 0.01					

5 Conclusion

The results support our hypothesis that the introduction of real-world case studies to an introduction to information systems course will increase student engagement. Class test exhibited a statistically significant improvement in self-assessed familiarity in six out of seven major learning components compared with the control group. The results for attitudes towards the profession were mixed. Students in the class test had greater statistically significant gains in five areas primarily related to enthusiasm for the subject and sharing it with others; however, other questions saw mixed or even decreased responses. A future line of inquiry could investigate the causes of these decreases and potential interventions to help students in an introduction to information systems course have a greater appreciation for various issues of importance to the technology profession. Additionally, HIPs are known to positively impact underserved student populations. We hypothesized that the incorporation of the case study HIPs into this introduction to information systems will lead to better outcomes for these students. We were unable to directly assess underserved student populations due to study limitations.

References

[1] Hart Research Associates. Fulfilling the american dream: Liberal education and the future of work: Selected findings from online surveys of business executives and hiring managers., 2018.

[2] Margaret E Beier, Michelle H Kim, Ann Saterbak, Veronica Leautaud, Sandra Bishnoi, and Jaqueline M Gilberto. The effect of authentic project-based learning on attitudes and career aspirations in stem. *Journal of Research in Science Teaching*, 56(1):3–23, 2019.

[3] Robert G Bringle and Julie A Hatcher. Implementing service learning in higher education. *The Journal of Higher Education*, 67(2):221–239, 1996.

[4] Jayne E Brownell and Lynn E Swaner. High-impact practices: Applying the learning outcomes literature to the development of successful campus programs. *Peer Review*, 11(2):26–31, 2009.

[5] Andrew Cox, Philippa Levy, Peter Stordy, and Sheila Webber. Inquiry-based learning in the first-year information management curriculum. *Innovation in Teaching and Learning in Information and Computer Sciences*, 7(1):3–21, 2008.

[6] Terrah J Goeden, Martha J Kurtz, Ian J Quitadamo, and Carin Thomas. Community-based inquiry in allied health biochemistry promotes equity by improving critical thinking for women and showing promise for increasing content gains for ethnic minority students. *Journal of Chemical Education*, 92(5):788–796, 2015.

[7] George D Kuh et al. Excerpt from high-impact educational practices: What they are, who has access to them, and why they matter. *Association of American Colleges and Universities*, 14(3):28–29, 2008.

[8] Tania D Mitchell. Traditional vs. critical service-learning: Engaging the literature to differentiate two models. *Michigan Journal of Community Service Learning*, 14(2):50–65, 2008.

[9] Brian K Payne, Lisa Mayes, Tisha Paredes, Elizabeth Smith, Hongyi Wu, and ChunSheng Xin. Applying high impact practices in an interdisciplinary cybersecurity program. *Journal of Cybersecurity Education, Research and Practice*, 2020(2):4, 2021.

[10] Bruce M Saulnier. Towards a 21 st century information systems education: High impact practices and essential learning outcomes. *Issues in Information Systems*, 17(1), 2016.

[11] Elaine Seymour, Douglass Wiese, A Hunter, and Susan M Daffinrud. Creating a better mousetrap: On-line student assessment of their learning gains. In *National Meeting of the American Chemical Society*, pages 1–40. Pergamon Amsterdam, 2000.

[12] Randy Stoecker, Amy Hilgendorf, and Elizabeth A Tryon. *The unheard voices: Community organizations and service learning.* Temple University Press, 2009.

Table 2: Responses for attitudes towards profession

Test-Pre	Test-Post	Diff.	Control-Pre	Control-Post	Diff.
I am enthusiastic about the course subject.					
3.76(1.03)	4.12(0.93)	0.36**	4.00(0.89)	3.86(0.73)	-0.14
I am interested in taking or planning to take additional classes in this subject.					
3.42(1.20)	3.91(1.16)	0.48**	3.43(0.87)	4.05(0.97)	0.62
I am willing to seek help from others (teacher, peers, TA) when working on academic problems.					
4.33(0.96)	4.15(0.94)	-0.18	4.24(0.94)	3.90(0.77)	-0.33
As business professionals, we are obligated to share our skills through outreach with our community.					
4.18(1.04)	4.55(0.90)	0.36*	4.33(0.86)	4.52(0.81)	0.19
It is critical that business students be able to communicate business concepts with fellow business students, business leaders, and other stakeholders.					
5.00(0.83)	5.09(0.80)	0.09**	4.95(1.02)	4.81 (0.75)	-0.14*
It is critical that business students be able to communicate business concepts with non-business audiences.					
4.76(1.03)	4.91(0.95)	0.15	4.76(0.89)	4.67(0.91)	-0.10
I am confident that I can do this subject.					
3.67(0.85)	4.51(0.91)	0.85**	4.14(0.57)	4.33(0.80)	0.19
How likely are you to share your expertise by volunteering in your community?					
4.30(1.13)	3.67(1.02)	-0.64**	4.43(0.75)	4.00(0.77)	-0.43*
When working in my field, I consider the end user to inform my project design and communication.					
4.13(1.07)	4.06(1.14)	-0.07	4.29(0.72)	4.24(0.83)	-0.05
When working in your discipline, it is critical to incorporate diverse perspectives from outside your field.					
4.64(1.05)	4.55(0.89)	-0.09	4.57(0.68)	4.14(0.79)	-0.43*
How likely are you to seek guidance from those outside your discipline to inform your decision-making process?					
4.09(0.91)	4.87(0.92)	0.78**	4.52(0.68)	4.57(0.87)	0.048
In our profession, it is critical that we serve the community by sharing our expertise.					
4.36(0.86)	4.36(0.99)	0.00	4.57(0.68)	4.33(0.87)	-0.24
* Significant to 0.1; ** significant to 0.01					

Closing the Gap: Building Internship Programs for Career Readiness*

Bilal Shebaro[1,2], Jacqueline P. DeMuynck[2], Charles R. Hauser[2,3], Andrea Holgado[2,3], Rebecca S. Thompson[2,4], and Paul J. Walter[2,5]
[1]Department of Computer Sciences
bshebaro@stedwards.edu
[2]Institute for Interdisciplinary Science (i4)
[3]Department of Biological Sciences
[4]Department of Chemistry
[5]Department of Mathematics
St. Edward's University
Austin, TX 78704

Abstract

The need for academic institutions and organizations to provide internship opportunities for their students is vital for student academic and career success, especially in STEM, given the market growth and competition in this field. Building strong connections with industry partners is essential for providing STEM students with experience working on industry-level products and research. This paper details our institute's summer internship program, shares some insight into our contracts with industry partners, and discusses how our summer interns are evaluated, selected, and matched with projects for their summer internship. We hope this paper will provide guidance for other academic institutions interested in developing a similar program to grant more students access to invaluable career development opportunities.

1 Introduction

As the 4th Industrial Revolution unfolds, collaborations between industry and higher education are becoming ever more critical to ensure that academic institutions are graduating students with the knowledge and skills to be successful and competitive in the workplace [3, 4, 5]. Student readiness can be enhanced via internships that rely on cross-sector educational and professional networks. In particular, internships provide an effective way to foster the inclusion of underrepresented students in computing and STEM fields [1, 2]. In these efforts, we established the National Science Foundation (NSF) funded (grant award No. 1832282) Institute for Interdisciplinary Science (i4) [6] at our university to support interdisciplinary research, training, and professional development among STEM faculty, staff, and students. One of our most effective initiatives is the i4 Summer Internship Program. This program utilizes cross-sector cooperative agreements with leading local and national STEM organizations to provide students with fully-funded internship opportunities. It supports underrepresented students pre- and post-internship and introduces them to industry-level applications and products. If given the proper financial support, i4's Summer Internship program is easily replicable across other universities and organizations. This paper will walk through i4's process of creating cooperating agreements with companies, evaluating applicants, preparing students for interviews, equitable hiring practices, weekly check-ins, and post-internship growth opportunities.

Our compensation strategy is the key to i4's Summer Internship Program's success. Participants in i4's Summer Internship Program are paid ($5,000 for full-time eight-week internships) solely through our institute's budget. By providing companies with no-cost interns, i4 can attract industry partners and unique internship experiences. Funding the internships creates highly marketable internship opportunities for our students, ensures they receive equitable compensation, and strengthens the Institute's connections and appeal among local industry leaders.

This paper is organized as follows: Section 2 describes the call for new industry partners and lists the major points of the cooperating agreements. Section 3 discusses the student application process, evaluating applications, and interviews with industry partners. Section 4 discusses the student hiring and onboarding process. Evaluation data collection and post-internship proceedings are discussed in Section 5, followed by our concluding remarks in Section 6.

2 Building Connections with Industry Partners

A successful internship program starts with building connections and co-signing cooperating agreements with industry partners. This section describes the "Call for Partners" that we use to invite new industry partners every year and provides insight into the agreements we develop for industry partners that standardize expectations around providing safe, meaningful work experiences for our students.

2.1 Call for Partners

The Call for Partners occurs in October through December prior to the internship period. Beginning with a description of our university and institute's mission statement, the Call for Partners explains the funding, modality, and timeline of i4's Summer Internship Program. Each year, this call is sent out to i4's established partners and is promoted on i4's website, social media, and LinkedIn accounts. The call is often sent directly via alumni or other personal networks, which has been particularly useful for forming local partnerships. The Call for Partners lays the foundation on which i4's cooperating agreements are built.

2.2 Cooperating Agreements

The i4 Summer Internship Program hinges upon a carefully crafted cooperating agreement with industry partners. The i4 team works with company representatives to get the cooperating agreements fully signed by January prior to the internship summer period. The University Risk and Compliance Office ensures our cooperating agreement contracts utilize language to create safe and meaningful working conditions for our interns. The i4 summer interns are designated temporary university employees, ensuring that students are protected by all federal and state laws while working off-campus at our industry partner's locations. These protections include but are not limited to Title IX, federal and state labor laws, and worker's compensation in the case of injuries during employment. Additionally, the agreement asserts that the industry partner will be solely responsible for ensuring a safe work environment in a commercial setting free from hazards. Also, the i4 interns follow the university's labor rules for all student employees. As stated in the cooperating agreement, "no student shall perform work in excess of twenty hours per week during the regular school year or in excess of forty hours per week during the summer term."

The cooperating agreements also restrict the power of the industry partner's non-compete requirements to avoid placing limitations on students' ability to seek similar work at different companies upon graduation. Because of this,

industry partners must provide any forms or agreements to be signed by the intern, allowing the University Risk and Compliance Office to review and dispute any clauses that put students at a disadvantage. The cooperating agreements safeguard student labor and ensure the students can pursue their career goals upon graduation.

The cooperating agreements not only place limitations and protections on student internships, but also define what the industry partners must provide to ensure a high-quality internship experience. The agreement states that industry partners are primarily responsible for the supervision of the student interns and the work performed by participating students, noting that an internal professional will provide supervision of the intern with expertise in educational and professional background in the field of experience. Students thus will be given consistent and applicable mentorship for their internship projects, strengthening their skills and building their professional networks.

Once the cooperating agreements have been signed by both parties (the industry partner and the university), the i4 team finalizes the list of industry partners for placing students interns over the summer. After the cooperating agreements are signed, we work with our partners individually to determine the number of interns they can support, the types of potential student projects they would have opportunities for, and what student expertise they are seeking.

3 Application Process and Student Selections

The i4 team initially screens applicants to create a short list of student candidates for each industry partner. Representatives from the industry partners then interview their applicants to create the best pairs between students, partners, and projects. This section details this process and the resources we provide our students to succeed in an interview.

3.1 Application and Applicant Evaluations

The summer internship application is open during February. The application consists of 20 questions, some with multiple components. Application questions gather information regarding student demographics, contact information, professional references, company preferences, resume, transcript, and several short answer questions. Critical questions include:

1. What are your post-graduation career goals? How would an i4 internship support you on the path towards those goals?

2. How would working for your selected companies align with your career goals and expectations?

3. Describe any past job experience or previous internships.

4. Describe any life experience or additional information that would distinguish you from other applicants. In other words, why should the i4 team select YOU?

Each member of the i4 team evaluates student applications using the rubric shown in Figure 1. For the transcript and previous job experience pieces, the i4 team prioritized underrepresented students by altering the expected rankings. Transcripts are scored 1 point for a $0 - 2.49$ GPA, 2 points for a 3.5+ GPA, 3 points for a $2.5 - 2.99$ GPA, and 4 points for a $3.0 - 3.49$ GPA. This allows i4 to provide crucial experiential learning and work experience to those who might not meet the criteria for other student internship programs. Additionally, i4's applicant rubric reversed typical scores for job experience, giving 1 point to those with "significant amount of prior relevant experience" and 4 points to students with "no prior relevant job or internship experience." By changing the typical scoring system, the i4 Summer Internship Program attempts to increase relevant work experience among underrepresented student populations.

Criteria	1 (Unsatisfactory)	2 (Average)	3 (Good)	4 (Outstanding)
Resume	Unsatsisfactory. Many of the required resume components are missing (personal information, educations, skills, experience, etc.). Key ideas are difficult to find. Frequent (more than 6) grammatical errors and typos. Not well organized and does not follow standard formats and organization.	Satisfactory. Lists some of the required resume components (personal information, educations, skills, experience, etc.). Key ideas may be difficult to find. Some (less than 6) grammatical errors and typos. Somewhat organized and may not follow standard formats and organization.	Up-to-date. Lists most of the required resume components (personal information, educations, skills, experience, etc.). Key ideas are relatively easy to find. Very few (less than 3) grammatical errors and typos. Well organized and follows standard formats and organization.	Exemplary and up-to-date. Lists all required resume components (personal information, educations, skills, experience, etc.). Key ideas are easy to find. No grammatical errors and typos. Clear and well organized. Follows standard formats and organization.
Transcript	Good Standing Overall GPA 0 – 2.49	Good Standing Overall GPA 3.5 & above	Good Standing Overall GPA 2.5 – 2.99	Good Standing Overall GPA 3.0 – 3.49
Post-graduation career goals	Goals may be superficial or missing. Connection to i4 may not be mentioned.	Vague goals, connection to i4 mission is somewhat evident but may be weak.	Inlcludes an overview of career goals. Makes clear connection to i4 mission.	Career goals are clearly written and explained. Makes explicit and strong connection to i4 mission.
Company selection aligns with student career goals & expectations	Reasoning may be superficial, unclear, or missing.	General reasoning towards their selections without obvious connection to career goals.	Reasoning towards selction aligns with career goals.	Student clearly demonstrates relevant reasoning for company selection with explicit alignment to career goals.
Life experience. Why YOU?	Life experience examples may be generic superficial.	General examples of life experiences that may or may not be relevant.	Student includes some life experiences examples that may be relevant.	Student has clear and specific examples of life experiences that are relevant to their goals.
Previous job experience	Significant amount of prior relevant experience.	Some prior relevant experience.	Minimal prior relevant experience.	No prior relevant job or internship experience

Figure 1: Rubric used in evaluation internship applications

3.2 Interviews with Industry Partners

The i4 team selects a pool of potential interns based on their scores and the availability of their preferred internship opportunities. Selected students interview with two or three industry partners. Each industry partner is provided

two more applicants than they can support, allowing them to rank their preferences and rule out any potential applicants.

Prior to interviewing with their assigned industry partners, the pool of potential interns must attend an interview preparation event hosted by the University's Career and Professional Development Office (CAPD). The event informs students on common interview practices and offers them a chance to ask questions specific to their situation. Additionally, the CAPD representatives invite students to schedule personalized mock interviews with their team.

After the students interview with their assigned companies, the i4 team and industry partners determine the internship assignments. Following their interviews, the company representatives send the i4 team a ranking of their interviewees. The i4 team then compares students' rankings across companies and selects based on company and student preferences.

4 Hiring and Onboarding

Since hiring processes vary greatly between institutions, this paper will focus primarily on onboarding. In April, i4 hosts a mandatory onboarding event for the selected applicants. During this event, representatives from the University Risk and Compliance Office educate students on non-disclosure agreements (NDAs), university protections, and identifying non-compete clauses. The CAPD representative gives students tips on networking and making the most of their internships. Finally, the i4 team describes the hiring process and the students' requirements before, during, and after their internships.

Alongside the regular university hiring documents, students are required to sign some additional documents. First, students must sign an internship agreement detailing all student requirements for their internship period, including filing evaluations and post-internship assignments. The other remaining documents are the company-specific forms. As agreed upon in the cooperating agreements, industry partners must send over their company-specific hiring documents for University review. Once approved by the University Risk and Compliance Team, these documents are shared with students for signature prior to their internships. After the necessary documents are signed and the student is hired, the i4 Summer Interns are officially ready to begin their internships on their assigned start dates.

5 Internship Evaluation and Post-Internship Proceedings

Students are asked to record and share data about their internships during the internship, including bi-weekly timesheets, weekly reflections, focus groups, and pre-, mid-, post-internship surveys. For the weekly online reflections, the

interns write responses to prompts about how the internship experiences relate to their course work or career goals. The pre/mid/post online evaluations ask similar but more detailed questions. The in-person focus groups only occur once towards the end of the internship period and allow the interns to provide greater detail on their internship experiences. These evaluations allow the i4 team to understand the students' experiences throughout the internship and are a part of the quantitative and qualitative reporting for the grant and to assess the measured benefit of the program to our students.

The i4 interns must complete several assignments after the internship. The interns meet individually with a member of the i4 team to review their experiences. This one-on-one meeting allows i4 team members to get a more in-depth understanding of what the students accomplished during their internships and what they liked and disliked about the program itself. These interviews provide the team with insight into improvements for the following year, usually reflected in timeline edits, application changes, or discussions with specific industry partners. In addition to the one-on-one meetings, students record an informational video to talk about their internship experiences, learnings, and tips for future interns. An edited collage of these videos is shared on our YouTube page for promotional purposes, especially for future interns. The one-on-one interviews and promotional video act as tools for information gathering, program development, and marketing when appropriate.

Finally, the i4 interns are encouraged to present posters at an annual school-wide symposium in the Fall semester following their internships. These posters allow the i4 interns to share their discoveries with other students and articulate their work experience much like they would in a future job interview. The students' supervisors from their industry partners attend, which furthers the interns' networking opportunities and strengthens i4's partnerships. Post-internship presentations have become a valuable activity for all i4 participants and help to disseminate internship information to others.

6 Conclusion

The Institute for Interdisciplinary Science (i4) Summer Internship Program provides underrepresented students with industry experience and career development opportunities vital for job placement in the 4th Industrial Revolution workplace. At this point, several of our summer interns have accepted post-graduation job offers with their assigned companies, and many have pursued jobs or research in similar fields. The success of i4's Summer Internship Program can be attributed to its payment model, carefully crafted cooperating agreements, inverted rubric for selecting potential interns, and ability to adapt to participant feedback. This internship program can be replicated at other

institutions if given the proper support, granting more students access to invaluable career development opportunities.

7 Acknowledgement

This material is based upon work supported by the National Science Foundation under Grant No. 1832282.

References

[1] Allison BrckaLorenz, Heather Haeger, and Christen Priddie. An examination of inclusivity and support for diversity in STEM fields. *Journal for STEM Education Research*, pages 1–17, 2021.

[2] Maral Kargarmoakhar, Stephanie Lunn, Leila Zahedi, Monique Ross, Zahra Hazari, Mark Allen Weiss, Michael Georgiopoulos, Ken Christensen, and Tiana Solis. Understanding the experiences that contribute to the inclusion of underrepresented groups in computing. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–9. IEEE, 2020.

[3] Antonella Petrillo, Fabio De Felice, Raffaele Cioffi, and Federico Zomparelli. Fourth industrial revolution: Current practices, challenges, and opportunities. *Digital transformation in smart manufacturing*, pages 1–20, 2018.

[4] Klaus Schwab. *The fourth industrial revolution*. Currency, 2017.

[5] Klaus Schwab and Nicholas Davis. *Shaping the future of the fourth industrial revolution*. Currency, 2018.

[6] St. Edward's University. Institute for interdisciplinary science (i4). `http://www.stedwards.edu/i4`.

Teaching Blockchain in Security*

Bilal Shebaro
Department of Computer Sciences
St. Edward's University
Austin, TX 78704
bshebaro@stedwards.edu

Abstract

The increased interest in using blockchain technology in real-world and virtual applications makes it a desirable topic for students to learn more about. The implementation details of such a technology includes some of the major topics that are usually covered in introductory computer security courses. Digital signatures and crypto hashing are among the taught topics that are applied in the blockchain technology and implemented in cryptocurrency systems such as Bitcoins and Ethereum. Having all these computer security topics combined in one application makes it attractive for course instructors to examine with students in order to improve their course learning outcomes. Therefore, this paper highlights on the importance of teaching the blockchain implementation details in a computer security course, and provides an example of a class-wide interactive student activity that mocks the blockchain implementation to enrich students with hands-on experience of these computer security concepts.

1 Introduction

Instructors of introductory computer security courses are eager to include more practicality while teaching their main course topics of cryptography (symmetric and public-key cryptography), hashing, access control mechanisms for authentication and authorization, software and network security, and so on. As learning by doing is one successful method in teaching such concepts, using

suitable real world practical examples that are also of high interest to students is crucial to improve the course learning outcomes. In this paper, we encourage instructors of such a course to explore with their students the blockchain technology and its implementation details in the cryptocurrency systems. Not only are such technologies of great interest to our students as they are shaping the future of many technologies and applications [2], but also the detailed implementation of the blockchain technologies involves a great application of digital signatures and crypto hashing. We also highlight the importance of teaching how Bitcoins [11] and Ethereum [4] (using non-fungible tokens (NFTs) as an example) work behind the scenes [12]. Doing so gives students an overview of how these systems work, and provides them with a better understanding of their capabilities and applicabilities, how to use them safely, their risks and limitations, as well as how to use them in various applications and domains. We finally provide an example of a hands-on student activity that instructors can explore with their students. This activity allows students to learn and mock the "behind the scenes" implementation of the mining process in blockchain, and gain hands-on skills in applying digital signatures and verification using PGP keys, and SHA-256 crypto hashing in real-world problems [8].

As this paper explores the importance of teaching blockchain and cryptocurrency among the topics taught in an introductory computer security course, the rest of this paper is organized as follows: Section 2 discusses the motivations behind teaching about blockchain technologies and what makes them suitable to explore in a computer security course. Section 3 includes some listing of topics in cryptocurrency implementations that directly link to the course key concepts. Finally, we share an example of a hand-ons class activity that mimics the Bitcoin blockchain system in Section 4, followed by our concluding remarks in Section 5.

As a disclaimer, while this paper encourages instructors to teach blockchain and uses Bitcoin and Ethereum as examples of blockchain cryptocurrencies, teaching students how to trade or invest in these digital currencies is out of scope and not endorsed in this paper.

2 The Motivation behind Blockchain

Blockchain technology was introduced in October 2008 as part of the Bitcoin cryptocurrency system that decentralizes digital currency and uses a peer-to-peer network to keep record of transactions and balances. It uses cryptography as a central part of its protocol, in order to establish pseudonymous and decentralized ledger transactions and currencies. Ethereum, Binance Coin, Tether, Solana, and others are some examples of cryptocurrencies that use the blockchain concept in their own modified versions. Ethereum, for example, uses

the concept of smart contracts that can define rules, like a regular contract, through submitting transactions to the blockchain. Bitcoin, for example, uses digital signature algorithms (same mathematical techniques used for a type of encryption based on elliptic curves) to sign, verify, and proof the user identity of transactions [9, 8]. In addition, it uses 256-bit SHA hashing algorithms (the same level of security that banks and the military use to encrypt their systems) to justify the concept of proof-of-work (POW), which is the key concept in trusting the transaction log of this currency. Hashing is used to also provide a unique fingerprint of every transaction, recording and authenticating them. The existence of these key concepts of computer security in blockchain and cryptocurrencies is what makes them an appropriate example for discussion in an introductory computer security course.

Additionally, instructors should strive to include examples and topics that are of high interest to student in order to increase engagement and positively affect the learning outcomes of the course. Given that Non-Fungible Tokens, or NFTs, was among the most searched items in 2021 on the Google search engines, it is currently of high interest to students [7]. As per an NFT industry report by market tracker DappRadar, in 2021, the NFT space generated over $23 billion in trading volume [5]. NFT transactions use the Ethereum cryptocurrency as the most suitable cryptocurrency for such an application, understanding how Ethereum works and the concept of smart contracts becomes relevant for students to learn how it works, and why NFTs can be done in Ethereum and not other cryptocurrency, such as Bitcoin. Such an explanation will not only enrich the students' understanding of computer security concepts, but it will also allow students to learn the reasons why some applications are suitable for one cryptocurrency over the other.

3 What to teach in Blockchain

Decentralized ledger system, online bidding system, Bitcoin cryptocurrency, and NFT (using Ethereum) are among the recommended topics that we suggest to explore in an introductory computer security survey course.

In a decentralized ledger system, financial transactions are stored in a chain of blocks. Each transaction is digitally signed and distributed among all network peers. Blocks are confirmed for their accuracy among the network peers through hashing and mining, an interesting application that combines key concepts in cryptography.

The study of how Bitcoin works is a relevant example to explain the key role that data miners play in the accuracy of this cryptocurrency. When a user creates a transaction, such a transaction gets digitally signed using the user's private key and is broadcasted to the Bitcoin peer-to-peer network users.

When we have enough transactions to form a block, data miners compete in trying to validate the block through calculating a certain hash function to solve the Proof Of Work (or POW - a consensus algorithm) concept. The use of hash functions, POW, and building a trusted chain of blocks that all miners agreed on its accuracy is the key success factor of the Bitcoin cryptocurrency. In Section 4, we discuss how such an algorithm is mocked as a classroom activity and how students are actively recording and mining transactions created among each other.

Another attractive example is the discussion of NFTs, ranked among the top searched terms in Google search engines in 2021, which allows users to create and trade digital items with differing values. NFTs have been a recent trend and students would be interested to learn about its relationship with the Ethereum blockchain, and why it would not be possible to be implemented in Bitcoin or other cryptocurrencies. On this matter, instructors could use previously taught computer security concepts to elaborate that every NFT is a unique token on the blockchain and cannot be replicated [10]. Each token is a digital certificate stored on a secure digital database that can be publicly verifiable as an intellectual property authenticated on a blockchain.

The reason why the Ethereum blockchain is a better suit to store NFTs than the Bitcoin blockchain is due to the need to store massive files on the blockchain. This is where the Interplanetary File System, or IPFS, comes into play as each NFT digital asset is stored with an IPFS hash into the smart contracts of the Ethereum blockchain [6]. The IPFS hashes are unique (because they are directly derived from the data itself) and can address large amounts of data, and the Ethereum blockchain stores these immutable and permanent IPFS links into its blockchain transactions. Instructors can dive into more details on how IPFS links work as a good hashing algorithm example, and how they can differ from regular http links. Such discussion allows students to understand the reason for using IPFS links over http links, and would be able to differentiate the types of applications that are suitable for each link type.

The discussion on how NFTs are designed and their purpose in providing a way to prove ownership of the work opens many doors for students' creativity. NFTs are designed to give users something that cannot be copied, which is the ownership of the work (though original owners can still retain the copyright and reproduction rights). This is because NFTs can be programmed to give royalties to its creator every time it is sold to a new owner. While NFTs are most known for selling and transferring ownership of anything digital (such as drawings, music, code, etc.), we are starting to see more applications in the digital and physical world that are being connected to NFTs. The idea that NFTs can store information about the ownership of a particular digital asset has also expanded to the physical assets [3]. NFT tokens have been

revolutionizing the ownership of tokenized art and intellectual property. For instance, one of the physical NFT service providers called MATTEREUM [1] defines a physical asset NFT as a unique digital token that denotes the right to take physical custody of an object, complete with warranties, insurance, and legal enforceability to create trust in trade. Such marketplaces are becoming more popular in digital assets, which are shaping a new crypto world of trading.

4 Blockchain in Practice and Student Engagement

Developing a hands-on activity to explain a theoretical concept is key in student learning. In the blockchain domain, we designed an in-class activity (which could also be assigned as a homework or project) that allows students to create a simplified version of the blockchain design in Bitcoin for the purpose of practicing the concepts of SHA-256 crypto hashing and digital signatures using the PGP public key algorithm.

The activity requires the use of a simple broadcasting messaging service (provided by the instructor) that listens and displays broadcasted messages to all network users. Students are given a unique public address (acting as their network identifiers), and are all assigned the role of miners. In addition, each student uses the GNU Privacy Guard software to generate their own PGP public/private key pair and make their public keys available to the entire class using the command:

gpg –keyserver keyserver.ubuntu.com –recv-key XXXXXXXX

where XXXXXXXX is the generated public key.

The activity consists of 4 steps:
(1)Transaction creation and signing
(2) Broadcasting
(3) Verification
(4) Validation

(1)Transaction creation and signing consists of student generated transaction messages. For example, when Student A is performing a transaction with Student B, Student B creates a transaction message and signs it with their private key. The message format looks like:

"A's public address, Amount, B's public address".

This message is then signed using the Student B's public PGP key and (2) broadcasted to the entire network of students in class. (3) These messages are signatures made with private keys, which can be verified by any student using

public keys.

(4) Validation in Bitcoin is performed using SHA-256 hashing for its Proof-of-Work (PoW). In our simplified version, all student miners record each broadcasted transaction into their ledger (or block), with each block can hold up to 10 transactions. Therefore as soon as we have a complete block (10 transactions), students will start mining their blocks while still listening to incoming transactions. We made mining the block a similar process to what happens in Bitcoin by requiring that the hash of blocks be with a certain predefined format, for example, a SHA-256 block hash that must start with 2 zeros. Block miners will need to try to find a random number that they can add to their block so that the hash of the block combined with that number can match the predefined hash criteria, mimicking what happens in Bitcoin hashing. This mining process will require a lot of hashing computations and the first miner who can find such a number and block hash value will announce it to the network. All other miners can easily verify the correctness of this number and the block hash value. The student miner of that block will be announced as the winner of mining that block. The activity continues in building more blocks, however, new blocks should start with the hash of the previous block to keep them in sequence, hence forming a blockchain.

This class activity will require students to write their own code to digitally sign transactions, encode them, and store them into blocks. Students would also use standard SHA hashing libraries for mining and computing their block hashes.

While the learning outcomes of such an activity directly match those outcomes in the typical Computer Security course, we plan to expand this activity in future iterations to make a simple trading website of digital artwork available to students with each picture represented by an NFT (with an IPFS link). Students could then implement a simple version of the Ethereum blockchain and design their own smart contracts that store the IPFS links and other owner information in their blockchain.

5 Conclusion

This paper highlights the benefits of teaching the blockchain technology and its implementation details to students in an introductory computer security course. These benefits come as a result of applying several key concepts of cryptography in blockchain examples, such as Bitcoin transactions and NFTs, which may be of high interest to students and highly relevant. While blockchain and cryptocurrencies can be their own course, this paper serves as a recommendation for general Computer Science curriculums that have an Introduction to Computer Security course as one of its courses without having Blockchain and

Cryptocurrencies as a dedicated course.

References

[1] Mattereum: Enabling trust for physical asset nfts. https://mattereum.com.

[2] BlackTie Academy. How NFTs are shaping our world. https://blacktieacademy.com/altcoins/how-nfts-are-shaping-our-world.

[3] NFTically Blog. Can I convert physical art into NFT? https://www.nftically.com/blog/can-i-convert-physical-art-into-nft/.

[4] Vitalik Buterin. Ethereum whitepaper. https://ethereum.org/en/whitepaper/.

[5] DappRadar. 2021 Dapp industry report. https://dappradar.com/blog/2021-dapp-industry-report.

[6] IPFS Docs. Mint an NFT with IPFS. https://docs.ipfs.io/how-to/mint-nfts-with-ipfs/#a-short-introduction-to-nfts.

[7] The Indian Express. Google trends data shows interest in 'NFT' is now greater than 'crypto'. https://indianexpress.com/article/technology/crypto/google-trends-data-reveal-interest-in-nft-is-now-greater-than-crypto-7689897.

[8] Azzief Khaliq. The good, the bad and the ugly of bitcoin security. https://www.hongkiat.com/blog/bitcoin-security.

[9] Bitcoin Magazine. Bitcoin security: Trustless private messaging with public and private key cryptography, 12 2021. https://bitcoinmagazine.com/technical/trustless-private-messaging-bitcoin-security.

[10] The Verge Mitchell Clark. NFTs, explained. https://www.theverge.com/22310188/nft-explainer-what-is-blockchain-crypto-art-faq.

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Bitcoin. org. Disponible en https://bitcoin. org/en/bitcoin-paper*, 03 2009.

[12] Gayan Samarakoon. Bitcoin fundamentals: Step by step explanation of a peer-to-peer bitcoin transaction. https://samarakoon-gayan.medium.com/bitcoin-fundamentals-a5d62fe98bac.

Computer Science Fundamentals Open Educational Resource: A Video-Based Practice to Learning Programming*

Christian Servin and Nadia Karichev
Computer Science & Info. Tech. Systems
El Paso Community College
El Paso, TX, 79915
{cservin1, nmerzlya}@epcc.edu

Abstract

Computer Science Fundamentals (CSF) courses have become a popular study area from K-12 to higher education levels (i.e., community and technical colleges, and four-year institutions). Different educational approaches have been proposed to disseminate concepts in these areas (traditionally through books and online platforms including wikis, websites, forums). Although several resources are available to assist students in learning tricks or "how-to" for specific items, some lack curricular guidance to lead to a constructivist learning approach. Some of the other available resources rely on a solid mathematical background, which many potential computer science students might not have, discouraging students from pursuing a computer science or programming field, particularly from a K-12 environment and community colleges. In this paper, we report the experience of deploying an Open Educational Resource (OER) used for the CSF in a community college that hosts early college high school, workforce, and transfer students as part of the teaching community.

1 Introduction

The Computer Science Fundamentals (CSF), a.k.a., CS-I, CS-II, and CS-III, belong to the first two years in computer science education. Nowadays, dif-

ferent institutions offer these courses (e.g., K-12, community college, four-year universities). Although many institutions might follow curricular guidelines, recommendations, and best practices in these courses, traditional books in CS might consider additional material and concepts that might be useful or, in other cases, redundant to courses depending on a variety of reasons (e.g., articulation agreements between local colleges/universities, recommendations from higher education authorities, specialized programs).

Furthermore, nowadays, books are designed with companions and other resources specified for specific courses, and instructors may or may not use them in their instruction, provoking students to purchase the entire book regardless of their decisions. Through the transition of face-to-face to online instruction due to the COVID-19 pandemic, many instructors revamped their teaching structure, style, and even the dissemination format, including resources and materials used at the "classroom." Codio, Zybooks, and other platforms gain popularity due to their auto-graded assessments and learning materials. Although these platforms provide valuable and relevant material for specific courses, many instructors considered these platforms complementary education resources for the material covered in the book or the class. Nowadays, vast online material (such as videos, wikis, question and answer knowledge market websites, tutorials). However, this material often does not follow a sequence of practice; it assumes specific background, knowledge, or skills before understanding a particular topic; the material is in a different programming language; is not aligned to a common curricular guideline recommendation/objectives. In this paper, we present *the CS OER: Open Educational Resource for Computer Science Fundamentals* designed for students in a two-year community college that hosts transfer students from a four-year college and high school.

This project was developed and implemented as part of the Open Educational Resources Course Development and Implementation Grant Program. The project received the award and funding from the Governor's Emergency Education Relief Fund (GEER).

2 Open Educational Resources and Computer Science Education

Open Educational Resources (OERs) are important in education, primarily in higher education levels, for several reasons:

- they are open licensed (permits that anyone can use the work, modify it, and distributed based on the discretion of the author)
- they are free and available for anyone (student/instructor/researcher)

depending on the available repository

- they are motivated to provide an alternative and enhanced educational paradigm from the current educational models.

These characteristics are significant and attractive to many small institutions such as community college or technical colleges that pursue similar objectives in their mission.
A Need for OER: In 2020, The Texas Higher Educational Coordinating Board (THECB) requested a solicitation to develop the Texas OER for different knowledge areas including. This was in response to the emergency commission department from the state of Texas. Due to the COVID-19 crisis and the abrupt migration to online classes, many institutions recognized the need to "revamp" many courses to online format. The Community College face relevant challenges than four-year institutions, such as:

- community colleges have a large non-resident population, this varies from international students to military students who take courses at different colleges while deployed.

- serve underserved student populations: including Early College High School (ECHS)/Dual Credit students who are traditionally underserved students, including those from low-income families, color, first -generation attend to college, and English language learners.

- assist part-time students with concurrent enrollment at a four- year college. Students who might take CS-I at the university level while completing their math requirements at the community colleges or taking summer courses at the college to speed up their graduation. These students might have had a book/resource at a different institution with different styles, emphasis, programing language, learning objectives, to mention a few.

- create roadmaps on the CSF courses: creating road maps with multiple "access points" permit to assist students from a different context to "go back" to specific areas and learn the competencies needed instead of taking a specific course or module. For example, a student who took CS-I class at a high school, and is currently taking a CS-II at a community college, might know the "why," however, details or semantics might be missing, and the student can find the "how" without excessively deviating.

- reduce/avoid cheating: Video watch can help decrease cheating or finding answers from the web and perform "copying and pasting." A student is likely to perform such behavior when the student is under pressure when he/she has the opportunity to do it, or rationalizes the opportunity

making the action least unethical [1]. OER does not necessarily prevent students from committing plagiarism but allows students to read, watch, and pay attention to the OER explanations that discuss the subjects explained in the course and not random answers from a search engine.

- achieve learning outcomes through videos: explaining how programs work from scratch requires a visual demonstration other than showing the code as the final result. Video watch can increase learning outcomes since it describes a step-by- step formulation of the problem and its solution as shown in various works such as [4].

2.1 Related Work

Although several existing OERs are designed for similar courses, there are no existing resources available for the particular characteristics mentioned in previous section. The work proposed uses a unique alignment to curricular guidelines that focus on the first two-year core in computer science. Although there are well-known best practices proposed in proceedings and NSF funded projects, this project considers the community college environment, including ECHS/Dual Credit High school, P-Tech courses, regular community college, and the first two-year material the regular four-year colleges. Finally, many different videos can be found on YouTube. However, there is no uniform in the material that has been presented, and the captions are accurate to the material presented. From the available open educational resources used for computer science education such as [3].

3 Proposed Open Educational Resource (OER) in Computer Science

The computer science fundamentals OER hosted at `https://www.thecsoer.com/`, offers three main contributions to the CS educational community:

- A mapping and alignment between the ACM curricular standards [2] and the Texas Essential Knowledge Skills (TEKS);
- A 50+ programming videos covering learning outcomes that incorporate the learning outcomes from the mapping; and
- A GitHub repository for each code presented in the videos

Additionally, the project provides detailed documentation for instructors, showing the mapping for each learning outcome. This instrument assists instructors in identifying and recognize potential material to be adopted into any

CSF course. GitHub allows students to "fork" a program and extended it based on instructions that the instructor would like to assign. The programs serve as a draft for future implementations.

Content, Topics, and Objectives: The OER consists of 14 Chapters (seven chapters for CS-I and seven for CS-II). Each chapter has three to five videos (depending on the complexity of the chapter). We identified topics from several CS-I and CS-II courses based on traditional courses and best practices in the computer science community.

Curricula Alignment: Currently, several curricular guideline recommendations are designed based on the educational level the institutions offer computing courses. For example, for higher education institutions (i.e., community and technical colleges, four-year universities) follow the ACM/IEEE curricular guidelines, wherein a K-12 environment might follow the curriculum provided by the College Board, CSTA, or code.org (among others).

Since community colleges share a large diversity of students, the design and implementation of the OER focus on serving these communities, including K-12 students, traditional community col- lege, four-year transfer, and particular workforce need for technical.

- **The ACM Curricular Guidelines**: This OER's content aligns to The ACM Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity. Most accredited four- year institutions worldwide follow these curricular guidelines as well-known standards to adopt in the undergraduate curriculum and keep consistent objectives.

- **The Texas Essential Knowledge Skills**: In Texas, K-12 schools follow the Texas Essential Knowledge Skills (TEKS). The three approved Texas Computer Science Courses that are commonly used in high schools are: Computer Science 1 (130.421), Computer Science 2 (130.4220), and AP Computer Science-A.

Coverage and Distribution of Learning Outcomes

Active faculty in curricular development and instruction studied the curricular recommendations and compiled the Learning Outcomes (LO) provided at these courses. These learning outcomes are associated with the contemporary topics offered in CS-I and CS-II. Next, they developed the video based on the LO selection. Figure 1 shows the total number of learning outcomes mapped from the four curricular recommendation and guidelines per course (CS1 and CS 2). The percentage indicates the LO covered per recommendation guidelines.

Observations and Challenges

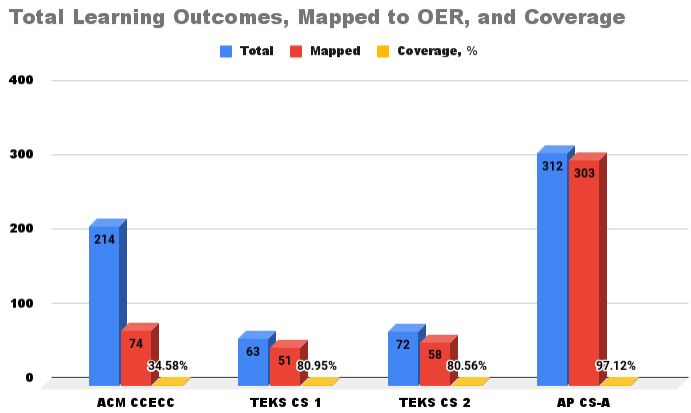

Figure 1: Mapping of Learning Outcomes from CS-I, CS-II based on ACM CCECC, TEKS CS 1 and CS 2, and AP-CS

- Selected ACM CCECC's KAs used in the FCS OER (with their respective coverage percentage) include the following: Software Development Fundamentals (94.74%), Algorithms and Complexity(82.35%), Programming Languages (50.00%), and Social Issues and Professional Practice (50.00%). Due to the nature of the FCS, a majority of the KAs had a lower coverage of LO, such as Computational Science, Information Management, Networking and Communications, Parallel and Distributed Computing, Platform-based Development, and Systems Fundamentals with a 0% of coverage.
- Exclusion of learning outcomes from the TEKS CS-I, CS-II, and the AP CS-A in the OER videos. These learning outcomes are overly specific, and it was a challenge to address to illustrate them. For example:

  ```
  CON-1.C.4 Values of type double can be rounded to the nearest integer
  by (int)(x + 0.5) or (int)(x - 0.5) for negative numbers.
  ```

  However, selected videos provide an "approximation learning outcome" along with a rationale to address its selection and replacement.
- Some learning outcomes have different levels of Bloom's taxonomy: the learning outcome inclusion is debatable. Depending on the context or the proficiency of instructors might select learning outcomes based on the cognitive level of Bloom's Taxonomy. For example, the learning outcome:

  ```
  130.421.c.6.l Understand the binary
  representation of numeric and non-numeric data in computer systems
  ```

was incorporated in Chapter 1, along with primitive data types. However, during the design of this work, the designers on the videos were considered to have more effective to illustrate this topic under Chapter 4: Functions/Methods.

4 Evaluation and Validation

The OER project consists of two evaluation components. Internal evaluation corresponds to the content and material evaluation through learning outcomes using controlled sections and external institution evaluations (project's partners). For the internal the project's administrators kept track of the following metrics to evaluate the usefulness and usability of the implemented system:

- Evaluation Instruments: A total of three tests; 10 quizzes that evaluate the material covered through each unit. A standardized (district-wide) final test for CS 1 and 2.
- Number of withdrawals or failing grades after using the OER project in CS 1 and 2.
- Compare students' saving costs for CS 1 students from the Spring 2021 semester vs. the Fall 2021 and the Spring 2022 semesters.

For the external project evaluation, the OER will be provided to leaders on partner institutions, along with a survey to evaluate the OER.

References

[1] Ibrahim Albluwi. Plagiarism in programming assessments: A systematic review. *ACM Trans. Comput. Educ.*, 20(1), December 2019.

[2] ACM Committee for Computing Education in Community Colleges (CCECC). *ACM Computer Science Curricular Guidance for Associate-Degree Transfer Programs with Infused Cybersecurity.* Association for Computing Machinery, New York, NY, USA, 2017.

[3] Beryl Hoffman and Barbara Ericson. *CSAwesome: A Free Curriculum and Ebook for Advanced Placement Computer Science A (CS1 in Java)*, page 1352. Association for Computing Machinery, New York, NY, USA, 2021.

[4] Colin Moore, Lina Battestilli, and Ignacio X. Domínguez. *Finding Video-Watching Behavior Patterns in a Flipped CS1 Course*, page 768–774. Association for Computing Machinery, New York, NY, USA, 2021.

Extended Precision Multiplication using a Message Passing Interface (MPI)*

Bill McDaniel[1] *and Evan Lemley*[2]
[1]*Department of Computer Science*
University of Central Oklahoma
Edmond, OK 73034
billmcd@roesner.us
[2]*Center for Research and Education in Interdisciplinary Computation*
College of Mathematics and Science
University of Central Oklahoma
Edmond, OK 73034
elemley@uco.edu

Abstract

This tutorial describes the Trachtenberg System [2] of multiplication using a Message Passing Interface (MPI) [1] running on multiple processors. Multiplication on a single processor is given first, then multiplication using multiple processors follows.

1 Single Processor Implementation

The essential logic of the non-MPI code is given as:

1. accept 2 strings (all digits)
2. make sure that they are the same length
3. convert from characters to numbers
4. do the multiplication
5. return the answer (as a string)

The original version of the multiplication routines were written as 2 functions in C++, and is shown below.

Listing 1: This is the main multiplication routine

```
string multiply(string a, string b)
 {
   int j,k, ls,rs;
   string ans = "";

   // make the strings equal length, jic
   while (a.size() != b.size())
     if (a.size() < b.size()) a = '0'+a; else b = '0'+b;

   // convert from characters to numbers so the
   // multiplications will work correctly
   int numdigits = a.size();
   for (int i=0; i<numdigits; i++) { a[i]-='0';b[i]-='0';}

   // do the multiply - 'ls' is the position of the
   // left side of the group to be multiplied,
   // and 'rs' is the position of the right side
   ls = rs = numdigits-1;
   int carry=0;
   do
     {
       for (k=ls, j=rs; k<=rs; k++,j--) carry += a[j]*b[k];
       handle_carries(carry,ans);
       if (ls > 0) ls--; else rs--;
     }
   while (rs > -1);

   // take care of any extra values in 'carry'
   while (carry) handle_carries(carry,ans);
   return ans;
 }
```

Listing 2: The following function converts the rightmost digit to a character and prepends it onto the answer then strips the rightmost digit

```
void handle_carries(long long int & carry, string & ans)
```

```
{
  ans = char(carry%10+'0') + ans;
  carry /= 10;
}
```

Notes:

1. The variable *ans* will contain the product when the algorithm is finished.
2. The *handle_carries* routine will pick off the rightmost digit, convert it to a character, then prepend it to the answer, creating the answer right to left.
3. A machine readable version of the code can be found at `www.roesner.us/~billmcd`
4. There may be a question about the maximum value for *carry*. Testing two 3000 digit numbers, we found that the maximum number for *carry* is 269990, so overflow is not a major concern.

This particular system for multiplication is not widely known, so, a specific example is shown below in Figure 1. The reader will note that the answer is created in reverse order (right to left).

2 Multiple Processor Implementation

Converting the above algorithm over to run using multiple processors is fairly straightforward. The approach used was to have one processor for each output digit.

The objective of this implementation was to have the calculations for various instances of t done at the same time, on different processors.

The MPI version of this code was run on the Buddy supercomputer at the University of Central Oklahoma. This resource was funded by a National Science Foundation grant (OAC 1429702) and consists of 31 general purpose nodes each with two ten-core Intel Xeon CPUs and 64 GB of RAM, and four high-memory nodes (identical to the general purpose nodes except for RAM size of 128 GB), and two GPU nodes each with one NVidia Tesla K-40 card.

Message Passing Interface (MPI) was introduced as a method of paralleliz-ing code in 1994 and has continued to be one of the primary ways code is run in parallel on high performance computing platforms.

The MPI functions that are used in the program are shown in Figure 2. Default values are used whenever possible.

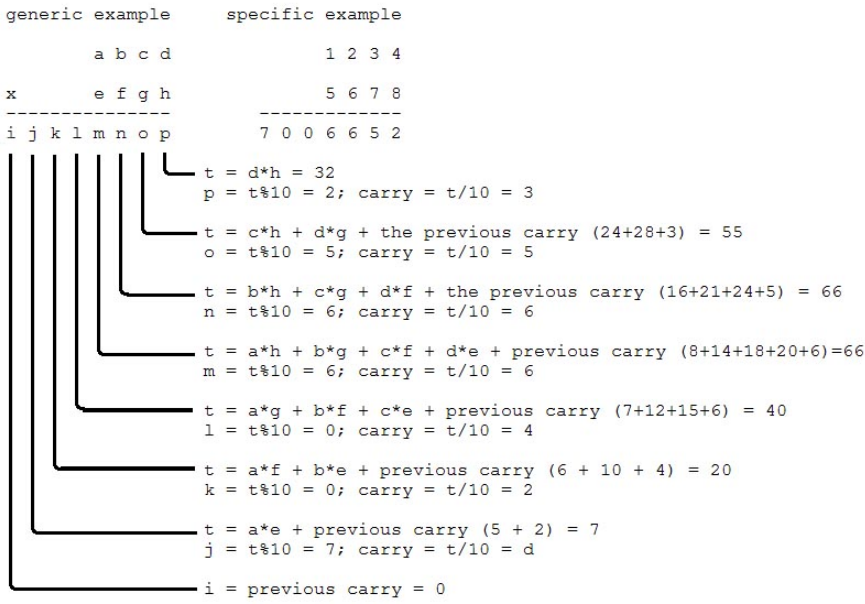

Figure 1: Given two 4 digit numbers that are to be multiplied together, where the digits are represented by letters, p is determined first, then o, then n, and so on. The variable t is a temporary variable to hold intermediate values.

```
The following MPI functions are used in the program.  Default values
are used whenever possible.

    // Initialize the MPI environment
    MPI_Init(NULL, NULL);

    // Get the number of processes
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    //                            └── a locally defined integer

    // Get the rank of the process
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    //                            └── a locally defined  integer
    // The variable 'my_rank' is a locally defined integer that is used
    // to identify each process.

    // send an integer
    MPI_Send(&rs,  1, MPI_INT, i, 0,  MPI_COMM_WORLD);
    //       │    │    │      └── 'to' process
    //       │    │    └── type of data being sent
    //       │    └── number of elements
    //       └── the data sent

    // receive an integer
    MPI_Recv(&digit, 1, MPI_INT,   i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
    //       │       │  │        └── 'from' process
    //       │       │  └── type of date being received
    //       │       └── number of elements
    //       └── the data received

    // this function terminates the MPI execution environment
    MPI_Finalize();
```

Figure 2: MPI functions used in the multiplication program

The program is basically divided into 3 parts.

1. if $world\_rank = 0$ then send the endpoints of the various groups to each processor and wait for the results to be returned.
2. if $(world\_rank > 0$ && $world\_rank <= N * 2 - 1)$ then accept ls and rs and do the multiplications.
3. if $(world\_rank = N * 2)$ then take care of the leftmost digit.

A quick summary of the logic of the program follows:

- Get the number of processes
- Get the rank of the process
- Get the name of the processor

pseudocode for process 0

```
determine the value of the left side and the right side
send the appropriate values of ls and rs to each process
    for (i=1; i <= N*2-1;i++)
        send ls to process i
        send rs to process i
        send a value of 0 (carry) to process 0
        if (ls > 0) ls--; else rs--;
receive the digits back (right to left)
    for (i=1; i <= N*2; i++)
        receive 1 digit
        result [N*2-i]= digit;
print the result - skip leading zeros
```

pseudocode for processes $= 1...N * 2 - 1$

```
get ls and rs from ps 0
do the computations
    for (k=ls, j=rs; k <= rs; k++,j--) carry += a[j]*b[k];
read tcarry from the previous process then add tcarry to carry
pick off the rightmost digit and send to ps 0
compute the new value for carry and send to the next process
    carry = carry \% 10
    send carry to the next process
code for process N*2
    get the carry from the previous process
    send the carry to ps 0
Finalize the MPI environment
```

Since the number of processes required has to be determined before the program starts, a shell script is used to determine some of the run time constants.

Listing 3: bash script to determine number of processes and other run-time constants before actual run begins

```
#!/bin/bash
#
echo "entering multiply.sh"
u=$1
v=$2
```

```
if [[ $u && $v ]]
then
  # determine the lengths of the 2 parameters
  lu=${#u}
  lv=${#v}

  # set lm to the length of the longer string
  if [ ${lu} -gt ${lv} ]; then lm=$lu; else lm=$lv; fi
  echo " -length of the longest string = "$lm

  # calculate the number of tasks needed
  p=$((lm*2+1))
  echo " -number of tasks "$p

  echo " -compiling"
  mpicc multiply.c

  # exit on compile time error
  if [ $? != 0 ]; then exit; fi

  echo "run the program"
  mpirun -np $p  "a.out" $u  $v   $lm
else
  echo  "usage: bash multiply.sh number number"
fi
```

Students in a senior level parallel programming course (or related) could be first asked to develop a single processor algorithm for Trachtenberg multiplication, then asked to develop a multiprocessor version of the single-processor code.

3 Acknowledgements

The code and scripts in this tutorial were developed and tested on the Buddy Supercomputer at the University of Central Oklahoma, which was funded by National Science Foundation grant, OAC-1429702.

References

[1] Message Passing Interface Forum. *MPI: A Message-Passing Interface Stan-*

dard Version 4.0, June 2021.

[2] Jackow Trachtenberg, as translated by Ann Cutler, and Rudolph McShane. *The Trachtenberg Speed System of Basic Mathematics*. Doubleday, Garden City, NY, USA, 1960.