Effective Techniques for Student Engagement in Introductory Computing Courses

S. Hussain1, S. Morgan2, and Z. Haque3
1Computer Science, Fisk University, Nashville, TN, USA
2Physics, Fisk University, Nashville, TN, USA
3Mathematics, Fisk University, Nashville, TN, USA

Abstract

We propose a four-objective plan to innovate the curriculum and the pedagogies for teaching and learning of computer science (CS) in order to create awareness, interest and success in CS as a discipline. The four objectives are as follows: a) Computational Thinking in CS courses for non-majors; b) Appealing Tools in Introductory Courses; c) Computing in Cognate Courses (Mathematics and Physics); and d) Utilize peer mentoring/coaching to foster deeper learning in related courses and to serve as ‘coaches’ in course-assigned projects. Due to these changes, the enrollment in CS increased by around 250% and students obtained competitive internships and job opportunities (e.g., Google, Microsoft, Apple, and Amazon). Students participated in hackathons and research conferences.

Keywords: Computational thinking, student engagement, pedagogy, peer mentoring.

1 Introduction

In this work, which is partially sponsored by NSF Grant# 1332432, we address the national problem of insufficient number of students enrolled in computing related disciplines. The U.S. Department of Labor predicts that there will be more than 1.4 million computing jobs available by 2020 [1]. Despite the increasing number of computing jobs, interest in these majors and careers has steadily declined over the past decade. Fewer students are enrolling in computer science and graduating with computer science degrees. If current trends continue, only 30% of these 1.4 million computing jobs could be filled by U.S. computing graduates by 2020 [1]. Further, the decline situation in women is even more significant. In 2010, women earned 57% of all bachelor’s degrees, yet they only earned 18% of computer and information science bachelor’s degrees, which is down from 37% in 1985 [1]. Moreover, there is 79% decline in the number of first-year undergraduate women interested in majoring in Computer Science between 2000 and 2011 [1]. Finally, it should be noted that the low percentage of women in IT industry has a significant impact on innovation and productivity. Perhaps the lack of women and minorities in CS is due to ‘stereotype threat’ [6], where students of color or women underperform academically because they harbor low expectations projected upon them by society and their previous instructors, particularly apparent in the area of Science Technology Engineering and Mathematics (STEM). Students who believe they will not excel in STEM develop behaviors that facilitate this outcome. Perhaps this explains why more than 50% of all African American professionals are graduates of HBCUs, where a nurturing and ‘safe’ cultural environment occurs as the backdrop for learning. Though HBCUs represent just 4% of US Universities, HBCUs confer nearly 35% of all Bachelor’s degrees to African-Americans in CS [3]. Cognitive researcher DiSessa postulates that our youth must have computational literacy to be active participants in the contemporary workforce [5]. Although computer literacy is more microscopic in concept, computational literacy is much more infrastructural. Computational literacy could be more likened to the ability to interact and develop systems, be them physical or procedural, to enhance one’s life in the workplace, classroom, or life’s challenges. DiSessa would suggest that computer technology can be a technical foundation of a new and dramatically enhanced type of literacy that will be as in-depth as and have as much influence as the emergence of text based literacy [5]. Others suggest that computational abilities are essential to being full participants in citizenship [7]. Many students who fall behind academically in mathematics and related curricula become counted out of society even before they leave school. Thus, it becomes essential for universities to equip its graduates with positive learning experiences that will enhance their computational abilities.

The national retention rate for computer science majors is only 20.6%, which is very low as compared to other majors: 56.7% Life Sciences, 57.9% Physical Sciences, 54.6% Mathematics, 44.6% Engineering, and 50.7% Social Sciences [2]. According to PCAST [2], “Merely increasing retention from 40% to 50% would translate to an additional 72,500 STEM degrees per year,
comprising almost three quarters of the 1 million additional STEM graduates needed over the next decade.” Most of the bright and potential students leave STEM programs after introductory courses because the courses are not very exciting or challenging [2]. Similarly, if students can successfully complete year one, their chances of degree completion are much higher [3]. Our four proposed strategies are based on the following two PCAST (2012, Executive Report, page x) recommendations:

- “Recommendation 1: Catalyze widespread adoption of empirically validated teaching practices.”
- “Recommendation 2: Advocate and provide support for replacing standard laboratory courses with discovery-based research courses.”

2 Methodology

Objective 1 - Computational Thinking in CS courses for non-majors.

Currently, non-major students have only one option (CS 100, 3 credit hours) to enhance their computation skills. However, there are several students, who are although proficient in general computing (e.g. word processing, presentations, computer organization, etc), would be interested in developing mobile apps and web applications. Due to free online tools and web services (e.g. Google App engine), it is possible for non-major students to develop web-based database applications. In addition to providing training for collaborative tools and cloud-based resources, the students are trained for computational thinking by developing mobile apps (AppInventor of MIT Media Lab) and writing scripts (Google’s App Script) in order to develop real life applications. Further, the concepts of word processing, data management, and decision support systems are provided for both PC-based (Windows and Mac) and web-based development environments (e.g., Google Drive).

We introduce two new short courses (CS 106 – Mobile Apps and CS 108 – Web Apps, 1 credit hour each) that would be beneficial to all the students. In current practice, several students take electives of 1-2 credit hours from Music, Arts, and other disciplines in order to obtain well-rounded liberal arts education. Consequently, the students obtain the opportunity to receive computation training and being equipped with general and exciting CS tools while having enough credit hours for their respective majors.

CS 106 – Introduction to Mobile Apps (1 credit hour) is a new introductory course (1 credit hour) in mobile apps and it is designed for non-majors and there is no-prerequisite.

Students use graphical tools to develop their mobile apps. This course attracts computer proficient students of other majors (e.g. Biology and Chemistry), who are even though proficient in traditional computing tools (word processing, presentation, web tools, etc.), they can benefit in creating their own mobile apps.

CS 108 – Introduction to Web Apps for (1 credit hour) is new introductory course (1 credit hour) for developing web apps. As web is the common platform for computers, smart phones, and various smart devices, it is beneficial to train students in developing web apps. The latest cloud-based services and collaborative tools are used in developing projects related to real life applications.

Objective 2 – Introducing Appealing Tools in Introductory CS Courses.

Overall, our goal is to take some of the computational tools and programming platforms that students will find ‘relevant’ and easily learned and focus on those in early courses in the CS curriculum pathway. Introducing visual programming and scripting languages in introductory CS courses on the pathway to the major is intended to effectively engage student interest in modern software development tools. In a pilot effort using this approach, freshmen students developed word games such as hangman, where they applied the principles of lists (arrays), lookup (linear search and binary search), sorting (selection sort, merge sort), shuffling, and animations (hangman costumes). Similarly, students used recursion to design graphical representation of fractals (trees and snowflakes), which would be very difficult to implement in traditional structures languages, such as C++. A significant literature affirms that integrating research into the curriculum stimulates greater student engagement, learning of context, and active integration of classroom knowledge into its confident use [8]. Thus, we provide a structure and funding to integrate research into students’ academic pathway in Computer Science. The goal of CS110, CS 120 and CS 241 is to foster critical thinking and an understanding of what various tools can and can’t do, while making the exercises stimulating, well-suited to group problem solving, and targeting a wide range of learning styles [4]. To test and improve this critical thinking, all of these courses have embedded projects in them, as part of the assignments that count to the final grade. Importantly, these changes in the way the courses are taught, and course-embedded projects in which students engage, allow participating students to behave as computer scientists, key to fostering their interest in this discipline for their career [9]. The expected outcome of these changes increase student continuation in each of the next courses in the curricular pathway. Summer internships are created to engage students in research problems whose results are presented in local, regional, and national competitive conferences for data dissemination and recognition of students’ research. Lopatto and others have provided data indicating that involving students in authentic research projects has a significant impact.
on underrepresented minorities, with a disproportionately greater positive impact on those currently underrepresented in the sciences, particularly African American males [10, 11].

Objective 3 – Computing in Cognate Courses

At Fisk University, the term ‘cognates’ refers to courses in other disciplines that are required in an academic pathway for graduation as a major; for example, in Computer Science, the cognate courses include mathematics (through Calculus II) and College Physics, a calculus-based course. One manifestation that students have successfully achieved a level of critical thinking efficacy is their ability to take concepts learned in one course and apply them in another. Consequently, we assure that this application of fundamental knowledge in another setting is expected and mentored by enhancing the cognate courses identified above with computer programming assignments and laboratories to foster computational thinking. As mathematics and physics courses are required for CS-majors, we infuse computational thinking and programming in Math 101 (College Algebra) 110 (pre-Calculus, including trigonometry), Math 120 (Calculus I) and in Physics 130 and 140. This integration of CS into these math and physics courses fosters the application of CS in new contexts for CS majors as well as encourage the introduction and use of computational tools and thinking for non-CS majors.

Objective 4 - Peer Mentors

A large literature affirms the positive impact of peer mentors in student learning, and have the reciprocal positive impact of training advanced and well-prepared students for effective communication and leadership skills, which are extremely beneficial for a successful CS academic and professional career. Peer mentoring fosters a scholarly environment for deeper learning and meaningful real-life applications across the four-year continuum of a student’s undergraduate experience [12, 13, 14].

We offer a professional development and leadership workshop before each semester for these students who have been selected to serve as peer mentors. The purpose of the training is to assure that the students are confident in their explanations of computer programs and computational tools, and also to help them learn how to foster discussions where questions are not directly answered, but are answered with questions that help the students come to their own understanding. Peer mentors commit 10 hours a week in this activity, including attending the didactic sessions with students to be better prepared to serve as cognate project mentors. This strategy is vastly different from that in individual student tutoring, which is more answer-oriented. Furthermore, the student peer leader engages all students in the discussion, so that the source of answer can percolate up from anyone in the discussion.

3 Results & Discussion

Due to these strategies, there was significant increase in enrollment and interest in CS courses. 333% increase in CSCI 110 enrollment (fifty students) compared to average of 5 years before the intervention. Similarly, 233% increase in CSCI 120 enrollment (twenty students) and 250% increase in CSCI 241 enrollment (fourteen students). Similarly, students obtained competitive internships at Google (16), Apple (2), Microsoft (5), Deloitte (8), Cisco (1), IBM (2), Amazon (3), and other national and regional competitive internships; moreover, students obtained full-time job offers from Google, Microsoft, Apple, Amazon, Cisco, Deloitte, etc. It should be noted that prior 2012, no student obtained internships at Google, Microsoft, Apple, Amazon, etc, and no student obtained full-time job offers at these competitive companies. However, in the last five (5) years, students job placement in competitive computing careers significantly increased. In addition to prestigious professional careers, students presented their research work at NCUR, Tapia, BKX/NIS, and other regional and national conferences as well as at Annual Fisk Research Symposium.

During grant period, additional computational physics exercises were included in the Physics 140 (University Physics II) curriculum. In 2015-16 academic year, the class had 14 students in the Fall semester, 40 in the Spring semester, most of whom were biology majors with a few physics, chemistry and CS majors. These exercises utilized MATLAB or GNU Octave. MATLAB is available in several Fisk computer labs, and links to GNU Octave downloads were provided for any students who wished to use their personal computers. A link for a web-based version of Octave (online http://octave-online.net/) was also provided. The first exercise was designed to introduce the students to the software and familiarize them with the MATLAB/Octave environment. Specific objectives of this assignment were entering commands into the command window and learning basic syntax, accessing ‘help’ and finding additional documentation, and creating simple scripts (.m files). Table 1 shows computing exercises developed for PHY 140.

Work is continuing on the Marble Game exercise because of the number of computational techniques it incorporates, its relevance to physics, and its relevance to the life sciences [15]. The marble game is a simple game that has been used to provide a conceptual framework for understanding a wide variety of physiological processes. Normally the game is used in a classroom setting with marbles and dice. The marble game is a simple Monte Carlo simulation of molecular partitioning between two compartments. The kinetics of this simple system can be used to model transport and
equilibrium for a wide variety of systems of physical and physiological interest including: membrane transport; diffusion; drug elimination; electrical conduction; osmosis; and ion channel gating. [http://circle4.com/biophysics].

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Physics concepts</th>
<th>Computational concepts/techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic plotting techniques</td>
<td>Create plots of several functions, label the axes and annotate the graphs</td>
<td>Create plots of several functions, label the axes and annotate the graphs</td>
</tr>
<tr>
<td>Kirchoff's laws</td>
<td>circuit analysis</td>
<td>Concept of arrays (matrices), additional matrix operations.</td>
</tr>
<tr>
<td>Graphing and fitting data</td>
<td>Analysis of experimental data</td>
<td>Arrays, visual display of data, data fitting</td>
</tr>
<tr>
<td>Equipotentials and Electric Field</td>
<td>Electrostatics, field and potential</td>
<td>Creating surface plots</td>
</tr>
<tr>
<td>Simple Differential Equations</td>
<td>Charging and discharging capacitors</td>
<td>Solving simple differential equations</td>
</tr>
<tr>
<td>Marble Game</td>
<td>Diffusion, electrical conduction</td>
<td>Loops, Intro to Monte Carlo techniques</td>
</tr>
<tr>
<td>Fermat's Principle</td>
<td>Refraction, Snell's Law</td>
<td>Data types, errors</td>
</tr>
</tbody>
</table>

Table 1. Computing Examples in Physics Courses

The math assignments related to computational thinking were assigned to students of Dr. Haque in his Calculus I (Math 120) and Pre-Calculus (Math 110) sections. There were 22 students in Calculus I (Fall 2015), 42 in Pre-Calculus (Fall 2015), and 20 in Pre-Calculus (Spring 2016). Most of these students are Biology majors with few Physics, Chemistry, Math and CS majors. Developed assignments are aligned to the objectives of these particular courses but reinforce students to think computationally, especially use of indexing, arrays, repetitions, and conditional statements in managing and manipulating data. Students were encouraged to use any programming language however emphasis was given to MATLAB. Table 2 shows a few sample computing exercises in Mathematics courses.

4 Conclusions

In summary, the proposed techniques show that effective curriculum modifications, course-embedded research projects, computing-based summer internships, computing in related courses (Mathematics & Physics), and peer mentoring support can provide significant increase in enrollment and student engagement for minority students in computing courses and consequently students success in academics and professional careers.

<table>
<thead>
<tr>
<th>Exercise</th>
<th>Math Concepts</th>
<th>Computational Thinking/techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plotting lines through two points</td>
<td>Slope and point slope form of straight line</td>
<td>Creation of linear space (array, x) and use of vector operation in a scalar manner to find array y. Then plotting y vs. x</td>
</tr>
<tr>
<td>BMI calculation</td>
<td>Evaluating Piece-wise function</td>
<td>Introduce conditional statement (if-else) and printf statement</td>
</tr>
<tr>
<td>Evaluating Trig functions</td>
<td>Use of Pythagorean formula of right triangle to find values of all trig functions</td>
<td>Use of variables (input and output) to evaluate expressions and printf output</td>
</tr>
<tr>
<td>Inventory management</td>
<td>Dot product of two vectors with any finite number of components</td>
<td>Storing data in arrays and retrieving data from array to compute element wise product in a for loop</td>
</tr>
<tr>
<td>Moving objects</td>
<td>Transformation of functions (Shifts, rotation, and reflection)</td>
<td>Introduction of 2D array in matrix multiplication</td>
</tr>
<tr>
<td>Solving system of linear equations</td>
<td>Matrix Algebra (inverse &amp; multiplication)</td>
<td>Interchanging two elements of a 2D array in computing inverse of a 2x2 matrix</td>
</tr>
<tr>
<td>Derivatives of discrete function</td>
<td>Using difference quotient to find 1st and 2nd derivatives of a discrete function</td>
<td>Creating arrays, storing values in an arrays and using these values in loop to compute derivatives at each point</td>
</tr>
<tr>
<td>Finding root of a differentiable function</td>
<td>Newton’s recursive formula for root finding uses 1st derivative</td>
<td>Unknown number of iterations require while loop and an appropriate stopping condition upon reaching a desired tolerance</td>
</tr>
</tbody>
</table>

Table 2. Computing Examples in Mathematics Courses

5 References


[2] PCAST (2012). "President’s Council of Advisors on Science and Technology (PCAST) -- Engage to excel: producing one million additional college graduates with degrees in science, technology, engineering, and mathematics, Report to the President".


