



Expanding capacity and promoting inclusion in introductory computer science: a focus on near-peer mentor preparation and code review

Heather Pon-Barry, Becky Wai-Ling Packard & Audrey St. John

To cite this article: Heather Pon-Barry, Becky Wai-Ling Packard & Audrey St. John (2017) Expanding capacity and promoting inclusion in introductory computer science: a focus on near-peer mentor preparation and code review, *Computer Science Education*, 27:1, 54-77, DOI: [10.1080/08993408.2017.1333270](https://doi.org/10.1080/08993408.2017.1333270)

To link to this article: <https://doi.org/10.1080/08993408.2017.1333270>



© 2017 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group



Published online: 07 Jun 2017.



[Submit your article to this journal](#)



Article views: 1975



[View related articles](#)



[View Crossmark data](#)



Citing articles: 5 [View citing articles](#)

Expanding capacity and promoting inclusion in introductory computer science: a focus on near-peer mentor preparation and code review

Heather Pon-Barry^a, Becky Wai-Ling Packard^b and Audrey St. John^a

^aDepartment of Computer Science, Mount Holyoke College, South Hadley, MA, USA; ^bDepartment of Psychology and Education, Mount Holyoke College, South Hadley, MA, USA

ABSTRACT

A dilemma within computer science departments is developing sustainable ways to expand capacity within introductory computer science courses while remaining committed to inclusive practices. Training near-peer mentors for peer code review is one solution. This paper describes the preparation of near-peer mentors for their role, with a focus on regular, consistent feedback via peer code review and inclusive pedagogy. Introductory computer science students provided consistently high ratings of the peer mentors' knowledge, approachability, and flexibility, and credited peer mentor meetings for their strengthened self-efficacy and understanding. Peer mentors noted the value of videotaped simulations with reflection, discussions of inclusion, and the cohort's weekly practicum for improving practice. Adaptations of peer mentoring for different types of institutions are discussed. Computer science educators, with hopes of improving the recruitment and retention of underrepresented groups, can benefit from expanding their peer support infrastructure and improving the quality of peer mentor preparation.

ARTICLE HISTORY

Received 9 July 2016
Accepted 11 May 2017

KEYWORDS

Peer code review; diversity and inclusion; peer mentors; self-efficacy; feedback

Introduction

Growth in career opportunities is predicted within computer science, engineering, math, and technology-related fields in the coming decade, and over 50% of the predicted STEM job openings are relevant to computer science (National Science Board, 2014). With greater awareness of the value of computer programming and technical competence for professionals across fields of study, colleges and universities are observing enrollment pressures for students who aspire to take a course to learn computer programming (NCWIT, 2013; Roberts, 2016). An important dilemma within computer science departments is to find ways to expand capacity within

CONTACT Heather Pon-Barry  ponbarry@mtholyoke.edu

© 2017 The Author(s). Published by Informa UK Limited, trading as Taylor & Francis Group.
This is an Open Access article distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives License (<http://creativecommons.org/licenses/by-nc-nd/4.0/>), which permits non-commercial re-use, distribution, and reproduction in any medium, provided the original work is properly cited, and is not altered, transformed, or built upon in any way.

introductory computer science courses that are both sustainable for the faculty and staff in the departments while promoting inclusivity and student retention. In this paper, we argue that a focus on feedback at the introductory level via peer code review, with assistance from trained near-peer mentors, can help departments meet these goals. When students engage in a core industry practice of peer code review, they also promote a community-oriented learning environment and strengthen their self-efficacy. We describe a model for preparing near-peer mentors, offer initial results of effectiveness based on survey data from the introductory students and the peer mentors, and consider ways other institutions may wish to adapt elements for their use.

It is well documented that pedagogy at the introductory level deters many students from pursuing STEM fields (Mervis, 2010). High failure rates within introductory computer science courses, in particular, are observed not only in the United States but also in Europe and Australia (Hoda & Andreae, 2014; Watson & Li, 2014). With the financial pressures facing higher education, increased enrollment demand does not automatically translate into additional faculty or staff. Enrollment pressures can create a catch-22 situation for computer science departments who lift the caps but in doing so, alienate learners from computing due to a lack of support in the curriculum.

Women, in particular, are severely underrepresented in computing. Since the 1980s, when the field observed an increase in the participation of women to over 30%, the percentages have declined to less than 20%, with African-American women and Latinas comprising less than 4 and 2%, respectively (Whitney, Gammal, Gee, Mahoney, & Simard, 2013). In prominent technology companies, the percent of women employed within their technical workforce is strikingly low (Business Insider, 2014). According to the National Science Foundation, National Center for Science and Engineering Statistics (2013), among first-year college students, only a very small number – less than 1% of women and less than 3% of men – intend to major in computer science. Furthermore, studies over the last two decades consistently demonstrate that while students from all backgrounds leave their computer science studies during college, women and underrepresented racial-ethnic groups leave at a higher rate (Beyer, 2014; Margolis & Fisher, 2002).

Computer science, thus, must take additional efforts to not only counter attrition, but also simultaneously change their culture and reputation in order to attract a greater diversity of students to the field (Whitney et al., 2013). A number of institutions have worked to do just that. For example, when Carnegie Mellon changed their recruiting and admissions procedures to reduce the emphasis on prior computing experience, they saw increases in women's enrollment (Margolis & Fisher, 2002). Indeed, given that male students were nearly four times as likely to take the AP Computer Science exam (National Science Board, 2014), this means computer science departments at large need to consider that the women who consider computer science as a major will often not possess experience. UC-Berkeley changed its introductory computing class – content and name – to emphasize the

beauty and humanity of computing, with great success at improving enrollments of women (Garcia, Chapman, Hazzan, Johnson, & Sudol, 2010). Harvey Mudd experimented with introductory courses organized by prior experience, so students with similar levels of prior experience were learning together (Alvarado, Dodds, & Libeskind-Hadas, 2012). Other institutions have experimented with pair coding (Braught, Wahls, & Eby, 2011), studio approaches (Reardon & Tangney, 2014), and peer instruction (Porter, Bailey Lee, & Simon, 2013). Computer science, as a field, still wrestles with negative stereotypes that deter participation from women and people of color (see also Cheryan & Plaut, 2010; Cheryan, Drury, & Vichayapai, 2013). By investing in a more collegial and supportive learning environment, computer science may become more attractive and better retain underrepresented groups, including women as well as men who do not thrive in the status quo environment.

Our work builds upon these efforts, seeking to address the dual challenge of providing a supportive, inclusive environment and yet manage the workload in sustainable ways. In particular, we raise the challenge of providing regular, timely, and specific feedback to introductory students, given that the first semester is when students' sense of potential competence is emerging and information (Kinnunen & Simon, 2012). While there are other valuable strategies, such as self-paced models, online instruction, and computerized autograding feedback, which also alleviate capacity concerns, we appreciate the approach of peer code review as delivered by near-peer mentors because this combination offers an investment in self-efficacy and community fostering attributes. First, we situate the current study within a socio-cognitive theoretical framework, explaining the importance of developing self-efficacy in the first semester of computer science coursework. Given the central role of feedback in self-efficacy, we articulate the ways in which peer code review can contribute to feedback. Finally, as a way to manage workload and invest in the development of student leadership, we describe the role that near-peer mentors can play, and why their preparation is worthy of investment.

Theoretical framework and literature review

Socio-cognitive perspective

This study is situated within a socio-cognitive model of learning, which emphasizes the complex interplay between an individual's self-beliefs, cognitive processes, and the learning experiences within one's social environment (Bandura, 1986). As human beings, we have the ability to self-reflect, take agency, and strategize about our learning. Self-efficacy is an important construct within this model, defined as a perception of one's ability to achieve in a particular domain (e.g. computer science), and strongly predicts an array of productive learning-related strategies and persistence in the face of difficulty (Schunk, 1991). While informed by prior experiences, self-efficacy can grow with incremental successes, through vicarious

learning from peers and constructive feedback from instructors; timely, ongoing, formative, and specific feedback is especially powerful (Pajares, 1996).

Feedback is also powerful when it communicates hopefulness about improvement, while still grounded in reality; women and Black students may be susceptible to receiving demotivating feedback, including “comfort feedback” which disingenuously reassures the student for lacking the skills (Rattan, Good, & Dweck, 2012), or critical feedback that targets only what is missing without any reassurance of a student’s ability to meet a higher standard (e.g. Cohen, Steele, & Ross, 1999). Establishing a situation where students receive ongoing, formative feedback means students will be more apt to gain insight into the limitation of their work with an eye toward improving that work over time (Shute, 2008).

As a self-perception, self-efficacy is not always accurate; students, particularly women, may underestimate their competency (Beyer, 2008). In addition, struggles with coding may threaten to minimize an emergent sense of self-efficacy. In a study of computer science majors at a large research university, Kinnunen and Simon (2012) found that even when students complete programming assignments successfully, they may retain a negative perception of their self-efficacy. Thus, initial programming experiences should have higher percentages of small successes in order to build self-efficacy, and students should gain feedback on their holistic programming experience, such as the concepts and process of the programming, not just summative feedback on the outcome (i.e. noting whether the program runs correctly). Along similar lines, Beyer (2014) researched the experiences of over 1000 first-year college students, finding that students who had a positive experience in their first CS course were more likely to take another course in the future. When instructors used pedagogically sound practices and students felt the assessment practices were fair, men and women alike were more apt to consider that a positive course experience. Given the importance of the first semester in a student’s emergent conceptions of competence and subsequent course-taking, the investment of support structures in the first semester is worthwhile.

Toward this end, working with another student through pair programming can offer a sense of community, a resource for improving self-efficacy, and greater understanding (Zarb & Hughes, 2015). In addition, pair programming has been appealing to female students (e.g. Liebenberg, Mentz, & Breed, 2012). One limitation of pair programming is that in some circumstances, pair programming can set up social comparisons that leave some students self-conscious or feeling less competent than their classmates (e.g. Kinnunen & Simon, 2012). To support pairs, research demonstrates that prepared scripts can help the pair to move through an obstacle and engage constructively with one another (Zarb & Hughes, 2015). One alternative or complement to pair programming is peer code review. Research suggests that peer code review can be less effective when the coders are not matched in ability or when one coder lacks enough experience to provide a useful review (Crespo, Pardo, & Kloos, 2004). As a result, we turn to look more closely at

the positives of peer code review and the advantages of having seasoned students to provide feedback.

Peer code review as a mechanism for feedback

In software development, peer code review is an industry standard process that not only strengthens code, but also saves the software industry thousands of dollars, as changes later in a development process are more costly to fix (Garousi, 2010). In peer code review, one coder works on programming, then talks with another programmer about the development process as well as the finished product (e.g. Demetriadis, Egerter, Hanisch, & Fischer, 2011). The code reviewer has a chance to describe observations and ask questions about the code that may lead to problems down the line, as well as provide encouragement and positive validation of effective code. Code review may take place among peers, between a supervisor and intern, or among groups of developers (Wang, Li, Feng, Jiang, & Liu, 2012). Peer code review can offer learners the chance to reflect on their coding; when conducted over time, the process can support the development of confidence in one's emergent competence (Demetriadis et al., 2011; Garousi, 2010).

Peer code review, as a pedagogical strategy to support the curriculum, has gained traction across U.S. universities, including Stanford University (Roberts, Lilly, & Rollins, 1995) and University of Arizona (Reges, 2003). Peer code review provides access to ongoing, formative assessment, where students are invited to participate in their own learning process as part of a community of learners (Søndergaard & Mulder, 2012). Rather than flag every potential error, code review offers a chance for mini-feedback sessions that focus on a core issue or concept that may be most helpful to the learner at that time. By working with a peer consistently, students may internalize the critical eye used by the peer review process, which in turn, improves their own self-evaluation and confidence in their abilities (Wang et al., 2012). This is especially beneficial for introductory courses (Hundhausen, Agrawal, & Agarwal, 2013).

For those who have used peer code review in the classroom, preparation for the peer code review process improves the experience. Demetriadis et al. (2011) scripted the process for peer code review and demonstrated that scripted groups invested more time in developing their solutions and deepened their general knowledge base more than those engaging in peer code review in a free-form way. Thus, we turn to the promise of training near-peer mentors to engage in peer code review, which has the potential to improve the quality of the feedback provided.

Investing in near-peer mentors: extend capacity to provide consistent feedback

The investment in near-peer mentors¹ is not new in STEM fields. As class sizes grow, faculty may find their ability to provide timely feedback is reduced. Near-peer

mentors help to make regular feedback a reality, and in doing so, provide support that can reduce attrition from introductory courses (Blanc, DeBuhr, & Martin, 1983). Working with peer mentors has more than the advantage of expanding capacity and support through the provision of feedback. By engaging in the field's practices, whether team-based problem-solving or code review within regularized peer-led workshops and mentoring meetings, the classroom becomes a community of practice. A community of practice promotes inclusion and belongingness; when newer students see themselves enacting the practices of seasoned students, they are more apt to see themselves belonging to the field and developing a professional identity (Lave & Wenger, 1991).

This is a seemingly subtle but important difference between tutoring and peer-led instruction. A tutor is reactive to the learner's questions, with the learner initiating interaction, often in response to a struggle. In contrast, peer-led instruction models are intentionally brought to students on a weekly basis, aligned with faculty instruction, to promote the disciplinary habits of practice. Two primary approaches to peer-led instruction are common in STEM fields: peer-facilitated study groups (Lewis, 2011; Treisman, 1992) and supplemental instruction (Peterfreund, Rath, Xenos, & Bayliss, 2008). Within engineering, math, and chemistry, facilitated study groups can be especially important because of each discipline's focus on problem-solving and to some degree team-based learning (Lewis, 2011; Liou-Mark, Dreyfuss, & Younge, 2010; Tien, Roth, & Kampmeier, 2004). Supplemental instruction, which involves a recitation of the lecture by a seasoned peer, and then focused practice on common misconceptions and difficult concepts, is common in many STEM fields such as biology (Rath, Peterfreund, Xenos, Bayliss, & Carnal, 2007) as well as in medical schools (Yu et al., 2011). Beyond the proactive, aligned nature of the interaction, peer-led instruction has an additional benefit in that it is intended to engage all students, not just those who are struggling (Liou-Mark et al., 2010). Peer-led instructional approaches have demonstrated robust benefits such as improved comprehension, grades, and pass rates in gateway courses, particularly for underrepresented students (e.g. Blanc et al., 1983; Lewis, 2011; Rath et al., 2007).

In an introductory computer science course, students are typically learning a new programming language. Part of the frustration new programmers face is the difficulty in figuring out what is wrong with a piece of code, as even an extra slash or period can lead a program to fail (Kinnunen & Simon, 2012). In this way, having a reader carefully examine an emergent coder's program can boost that coder's understanding and emerging sense of capability. An introductory computer science course shares attributes with an introductory writing course, where peer review and revision are both essential for grammar as well as style and organization (Chisholm, 1991). In computer science, learners benefit from critical thinking and conceptual depth, and this can be facilitated more easily with regular feedback from a near-peer mentor who is well positioned to encourage and invite newer

learners into the field. We anticipate that peer code review will only continue to grow as more computer science departments shift the reputation of their learning environments from solitary learning to a community of practice; learning more about the preparation of peer mentors is crucial to create a sustainable and effective practice.

Peer mentors need to be prepared for their role. Training courses are associated with the most effective models of peer mentoring, whether in Chemistry (Tien et al., 2004), Biology (Streitwieser & Light, 2010) or Engineering (Anagnos, Lyman-Holt, Marin-Artieda, & Momsen, 2014). This training provides an opportunity for peer mentors to improve their knowledge base, which helps to create a sense of credibility for peer mentors (Packard, Marciano, Payne, Bledzki, & Woodard, 2014). To support the peer mentors while they are in their role, they may be eligible for additional course credit (Streitwieser & Light, 2010) or in other cases, hourly pay or semester-based stipends (Bowling, Doyle, Taylor, & Antes, 2015). In most instances, the peer mentors meet regularly with the course instructor or otherwise gain access to the instructor's weekly goals through an ongoing seminar for peer mentors.

Even though they have distinguished themselves as effective learners, peer mentors are not automatically effective or inclusive teachers. Furthermore, peer mentors are not necessarily going to be viewed as approachable, creative in their teaching approaches, or otherwise contributing to an inclusive learning environment (Streitwieser & Light, 2010; Tien et al., 2004). Through their training, peers may need to learn to develop ways to adapt their explanations to benefit different learners and go beyond content knowledge explanations to deeper modeling and facilitation of problem-solving processes (Bowling et al., 2015; Tien et al., 2004). The research base documents that near-peer mentors become ambassadors for their departments, gain communication skills, strengthen their own knowledge, and deepen their commitment to the field (Anagnos et al., 2014; Bowling et al., 2015; Tien et al., 2004).

Current project

In the current project, we describe the near-peer mentor preparation and examine the perceptions of effectiveness of the peer mentoring, with input from the peer mentors and the introductory CS students. We ask:

- (1) Which components did the peer mentors find to be most effective in their preparation? How effective did they feel? What else would have helped them?
- (2) How did the CS1 students perceive the effectiveness of the peer mentors?
- (3) What did the peer mentors learn about the needs of CS1 students from their participation in this role?

Method

Context

This peer mentoring training course was developed at an all-women's liberal arts college in New England in the United States. The need to recruit and retain women in computer science is internationally recognized. Indeed, although this project is located at a small college, computer science at our college has grown to over 100 majors and minors, a similar number of women one would find at a large research university ten times our size. The college enrolls approximately one quarter international students, one quarter domestic students of color, and one quarter first generation for college. In addition, a number of students at the college, including students enrolled in CS1 or serving as peer mentors, do not identify as women, self-describing as transgender, genderqueer, or men.

Prior to our work to infuse peer code review via near-peer mentors, we were proud of our collaborative learning environment, exhibited by (1) an active student-led organization which sponsors a co-curricular mentoring program, hackathons, and career panels, as well as (2) embedding teaching assistant hours within a studio model on most afternoons and evenings. Creating an environment that is perceived as inviting and supportive is important to us, not only because we know that women are underrepresented in computer science, but also because we think this is the kind of environment we ourselves as faculty and staff would like to work within. We hope to attract a more diverse group of students with respect to race, national origin, class, and other social identities, to study computer science.

Like other institutions, enrollment pressures have increased within our introductory computer science sequence, with consistent waitlists each semester in recent years. Through an award sponsored by a large foundation, we were able to develop a curriculum for near-peer mentor training and incorporate peer code review into our introductory curriculum with a goal of expanding our enrollment capacity while keeping our commitment to inclusive practices. We recognize that we have been fortunate to retain small class sizes more typically observed in the humanities, with a cap of 18 students per lab section, with each student having access to a lab computer. We expanded to a semester class size of 72, with two lectures at 36 students each. The course covers what one typically finds within introductory computer science, including variables, functions, scope, conditionals, loops, control flow, abstraction, and arrays, with programming assignments due each week. We see the approaches described here as adaptable to large universities as numerous institutions facilitate study groups in other subjects, including chemistry, with over 100 students at a time, in flexible learning spaces (Griswold, 2010).

In the Fall 2015 semester, 71 students were enrolled in CS1. Among the 51 students who completed the survey, 53.0% intended to major in computer science, 23.5% planned to minor in computer science or take at least one more course, 15.7% were unsure about taking an additional course because they felt it was too late to major or minor, and 7.8% did not plan to take more computer science.

Peer mentor design

Peer mentors were assigned to a cluster of introductory students, at a 1:9 ratio. The expectation was for the peer mentor to provide written peer code review feedback on submitted assignments to each of these nine student mentees, using a code review tool (Upsource by JetBrains) and meet individually with each student, weekly, to review the feedback in person for approximately 10 min. Overall approximately five mentees consistently missed the sessions with the peer mentors, despite knowing attendance was factored into the final course grade. Meetings often took place in the science building or library, with an occasional meeting taking place in a dorm study area.

The weekly peer mentor meetings opened with the peer mentor asking the introductory student if there was anything in particular the student would like to discuss, based on the written peer code review feedback or about the course in general. Students who were excelling were encouraged to continue their good work, offered code style or program organization feedback, and asked if they had other topics for discussion; in some cases, students asked about internships. Students who were struggling were asked to talk through their process, to focus on one particular aspect of their code for the coming week, and encouraged to persist. Two excerpts from written code reviews are included in Appendix 1.

A subset of peer mentors also offered a weekly active learning workshop that corresponded with the week's material. Because of scheduling, not all peer mentors offered these and not all students could attend, and those data are not of focus in this report, although briefly mentioned. The subsequent iterations of the course have embedded these active learning workshops into an extended laboratory session each week to maximize the attendance of students, and to increase curricular-relevant contact with peer mentors and students each week.

The weekly time commitment for peer mentors was approximately 9 h per week: 75 min in practicum meeting with instructors and other mentors, 210 min on written code reviews (30 min/mentee), 90 min on individual mentee meetings (10 min/mentee), 120 min on either active learning workshop and preparation or assisting in the weekly lab section, 30 min on scheduling and logistics. The first cohort of peer mentors received course credit. In subsequent iterations, peer mentors received a combination of wages and course credit.

In this model, the peer mentors complemented the existing teaching assistant structure. Whereas teaching assistants held drop-in evening help sessions to aid students *prior* to submitting assignments, the peer mentors provided personalized, timely feedback to encourage reflection and deepen understanding *after* the assignments were submitted (weekly meetings typically fell 5–10 days after assignments were submitted). A CS1 student might receive help from several different teaching assistants during a semester. In contrast, with the peer mentor program, each CS1 student had a single peer mentor who reviewed their code and met with them weekly for the whole semester. While peer mentors did not

grade assignments, their feedback was referenced by instructors who performed the grading.

Peer mentor preparatory curriculum

The pre-training involved seven sessions (see Table 1). The first cohort of peer mentors enrolled in a summer bootcamp version of the training over two consecutive training weeks (10 days); subsequent training has been offered as a half-semester course over the span of seven weeks.

The curriculum was developed to focus on the three key aspects of the peer mentor's role: providing effective feedback in peer code reviews, meeting one-on-one for weekly coaching with the peer code review as a starting point, and active learning workshops. Critical to the training was the discussion of inclusive pedagogy throughout the training and practicum. This involved: emotional intelligence, inclusive pedagogy including gender and race influences on teaching, learning, and feedback, as well as research on active learning in STEM.

Students participated in a mock face-to-face feedback session which was video recorded and then reflected upon, a common practice within mentor training and promotion of reflective teaching more generally (see Tripp & Rich, 2012 for a review). In this training, they watched their own video recording privately first. Then, with a pair or trio, they watched excerpts of videos from their peers, scanning for helpful suggestions they could make to improve the practice, or take away lessons they could take to improve their own practice. To further the reflective practice, beyond the pre-training, during the first semester of their role, the peer mentors met weekly for 75 min as a peer mentor cohort with one faculty member from the department and one lab instructor to discuss the upcoming week's curriculum, to share successes, and to strategize about any emergent issues with particular students.

Table 1. Training curriculum by session.

Session	Topic
1	Diversity in CS Peer Mentor Role Expectations
2	Self-Efficacy; Self-Regulated Learning Reviewing Technical Topics: Types of Bugs Peer Code Review Tutorial
3	Peer Mentoring Roles; Effective Feedback Emotional Intelligence Written Code Review Practice
4	Inclusive Pedagogy; Impact of Race and Gender on Feedback Feedback Practice; Paired Mock Sessions
5	Best Practices in Active Learning Pair Active Learning Brainstorming Active Learning Pitches
6	Active Learning Review Session Dry Runs Mock Sessions: Reflections and Case Scenarios
7	Active Learning Review Session Dry Runs Q&A Conversation with current Peer Mentors

Methodological approach and data sources

A mixed methods approach, using a combination of quantitative ratings and qualitative, thematic analysis of participants' responses was used. The validity of qualitative research can be enhanced by cross-checking multiple data sources for convergence of ideas (Burke Johnson, 1997). We asked peer mentors to share their insights and experiences, an approach similar to other peer mentor studies in STEM (see Tenenbaum, Anderson, Jett, & Yourick, 2014). The ratings from introductory computer science students added to the perspectives offered by the peer mentors. We chose weeks 4 and 7 of the semester to ask the perception of effectiveness questions because we anticipated that the peer mentors would be far enough along into their role where a self-rating or student rating would be meaningful, and not so close to the end of the semester that the ratings would be detracted by end-of-semester business. Appendix 2 lists the questions for both peer mentors and CS1 students.

Peer mentors

Peer mentors completed surveys before and after their training to assess their own confidence and learning, as well as check-in questions requested during the practicum course (approximately week 4 and 7). We asked the peer mentors (1) what kinds of stumbling blocks are the CS1 students facing, (2) what advice can you offer the students to improve their work?, and (3) what would help you to be more effective in your role? They were also asked what was frustrating them and what was going well.

To assess their perceptions of their own effectiveness, we used DeChenne, Enochs, and Needham's (2012) scale of teaching efficacy, which was developed for graduate-level science teaching assistants. In this survey, computer science peer mentors rated their own perceptions of effectiveness in an array of areas. We focused on a subset of items most relevant to the peer mentoring role, reformulating the effectiveness items into three ratings: I am knowledgeable, I am approachable, and I use flexible/creative approaches. This allowed us to ask directly about these three elements, and facilitated the comparisons of the self-perceptions with the introductory students' perceptions by asking the same questions. Peer mentors completed these ratings at the end of the practicum sessions.

Introductory students

We used the three items for introductory computer science students to rate the effectiveness of the peer mentors at the same points in time (my peer mentor is knowledgeable, my peer mentor is approachable, and my peer mentor uses flexible/creative approaches). We also added four items that assessed the perceived contribution of the peer mentor feedback to their confidence and understanding.

We asked students to complete these ratings at the end of their lab sections using an on-line form; we secured participation from approximately half of the

students. The range of students who submitted ratings were diverse, to include those excelling and struggling, those who were planning to major and those planning to complete just this one course. As there were no differences in the ratings by the week requested (4 or 7), we combined the ratings for reporting purposes.

Peer mentor characteristics

The first group of peer mentors, consistent with the next two iterations of training, were juniors and seniors with at least two courses of computer science. Of the eight peer mentors trained in the initial cohort, three identified as biracial, people of color (two African-American/White, and one Asian-American/White), whereas the remaining five identified as Caucasian/White. In addition, six mentors identified as women, one as a man, and one identified as gender-queer. The peer mentors were recruited directly by the faculty in the department. All were CS majors, and several were double majors in subjects such as Math or Psychology. All expressed some confidence in CS material, although this varied across mentors; a couple of peer mentors felt more confident in their mentoring skills and felt less confident they were strong in all aspects of the curriculum.

Only one peer mentor was considering a career as a university professor with any professional interest in combining research and teaching; the peer mentors were more interested in careers as software developers or as consultants in industry and nonprofit organizations. Recommended to the position by their faculty members, the peer mentors already possessed an impressive array of internships with recognized industry leaders and with research in higher education, and in three cases, had already served as a teaching assistant. Their main motivation to sign up for the role was to help other students to improve their coding. They also wished they had had a peer mentor who provided peer code review when they had started, who would get to know them and provide regular feedback throughout the semester. Mentors received credit for the first semester of their practicum. They varied in their desire for pay or credit, as some students held other part-time jobs at the college. Currently, we are exploring the possibility of a mixed credit/pay structure.

Results

Peer mentors' reflections: preparation and effectiveness

Preparation

At the end of the practicum course, after completing one semester of mentoring, peer mentors were interviewed and asked what was most helpful about their preparation course. Peer mentors emphasized the practice they obtained was very valuable. They especially valued the videorecorded mock session and reflection, even though they were reluctant to engage in this part of the training at the

time. In particular, being able to watch their own recording in private helped to minimize any potential feelings of embarrassment. Watching excerpts from their peers' mock sessions, in a collaborative and constructive format, was helpful for sharing strategies.

- As much as I hated it, the mock one-on-one meetings [were most helpful]. The practice was great, and I feel more comfortable because I know that my first meeting will not have been with a real student.
- I'm glad we got to watch other people's mock one-on-one and discuss some of the strategies that seemed effective.
- Getting to watch ourselves helped with the reflection. I was able to see what I was doing well and what I could improve on... Now I know what it's like to have my code reviewed by a peer.

In addition, the discussion of inclusion was helpful because although they were aware that the field faces challenges with recruiting individuals from diverse backgrounds, they had not read original research literature or explicitly discussed strategies to help. By reading the literature and practicing what they read about within mock sessions, they could tune in more closely to the language they were using and also discuss more openly any stereotypes or assumptions they may hold about computer science learning. They also reflected on being more intentional in their feedback, to recognize and build on students' strengths, and help direct the students' energy constructively.

- The part of the training course on how diversity is affected by feedback was really interesting. It was interesting to see in a specific example how my own actions can affect someone's perspective in a way I hadn't considered before.
- Learning about how to give buffered feedback, especially balancing candid critiques and constructive praise, was very helpful.

Across the group, the peer mentors emphasized how valuable it was to be within a community of peer mentors so they could learn from one another.

When asked what would have improved their preparation, the peer mentors emphasized two points. One, they wanted a closer read of the CS1 faculty instructor's lesson plan, particularly as the instructors for the introductory course change each semester and go at a different pace, and two, they wanted strategies to manage situations when the mentees did not show up for individual appointments.

Perceptions of own effectiveness

Peer mentors were asked at two points in the semester about their perceptions of their own effectiveness. Looking at the items from the teacher efficacy scale (see Table 2), we can see that the peer mentors, on the whole, improved in their confidence on the range of items, whether getting students to ask questions, treating students with respect, or providing detailed feedback on their progress.

Table 2. Comparison of mentor self-reported teacher efficacy items (from DeChenne et al., 2012) pre-training and post-practicum; ratings range from 1 (Very Unconfident) to 6 (Very Confident).

Mentor	Time	Participa- tion	Invest- ment	Climate	Ask Qs	Encour- age	Respect	Feedback
1	Pre	4	4	5	4	6	5	5
	Post	4	5	5	5	6	5	5
2	Pre	3	4	4	5	5	5	5
	Post	4	5	5	6	6	6	6
3	Pre	6	6	6	6	6	6	6
	Post	6	6	6	5	6	6	6
4	Pre	4	5	5	5	5	5	3
	Post	4	6	5	5	6	6	5
5	Pre	6	3	5	3	5	5	4
	Post	4	3	4	6	6	4	5
6	Pre	6	6	6	5	6	6	5
	Post	5	5	5	5	6	6	5
7	Pre	5	5	5	5	5	5	5
	Post	5	6	6	5	5	5	6
8	Pre	3	4	4	5	5	5	4
	Post	5	4	4	5	4	5	5

Although the peer mentors arrived with confidence, the level of confidence varied from component to component, and from mentor to mentor. Consistently, the levels of confidence improved with training and experience in the role.

In addition, when asked about how knowledgeable, approachable, or flexible they were in their instruction, these self-perceptions also reflected improved confidence about their knowledge in particular. For example, when asked in week 4 whether they were knowledgeable, 37.5% slightly agreed (whereas the other 62.5% agreed or strongly agreed). In contrast, during week 7, 100% agreed or strongly agreed. In week 4 and week 7 alike, they agreed or strongly agreed they were approachable. And in week 4 and 7, 75% of the peer mentors agreed or strongly agreed they were creative and flexible about developing solutions with 25% slightly agreeing. We note later that the self-perceptions are supported by the introductory students' ratings of these same qualities.

Ratings of peer mentor effectiveness by CS1 students

When rating peer mentors, the CS1 students consistently rated the peer mentors as knowledgeable, approachable, as well as flexible and creative in their teaching approach, as shown in Figure 1. A 6-point scale (1 = Strongly Disagree, 6 = Strongly Agree) was used for students to rate their level of agreement with a series of statements (included in Appendix 2). A few students added optional comments. For example:

- All of our in-person meetings have been very helpful and informative.
- I really enjoy meeting with [my peer mentor] and have nothing but positive things to say about them.

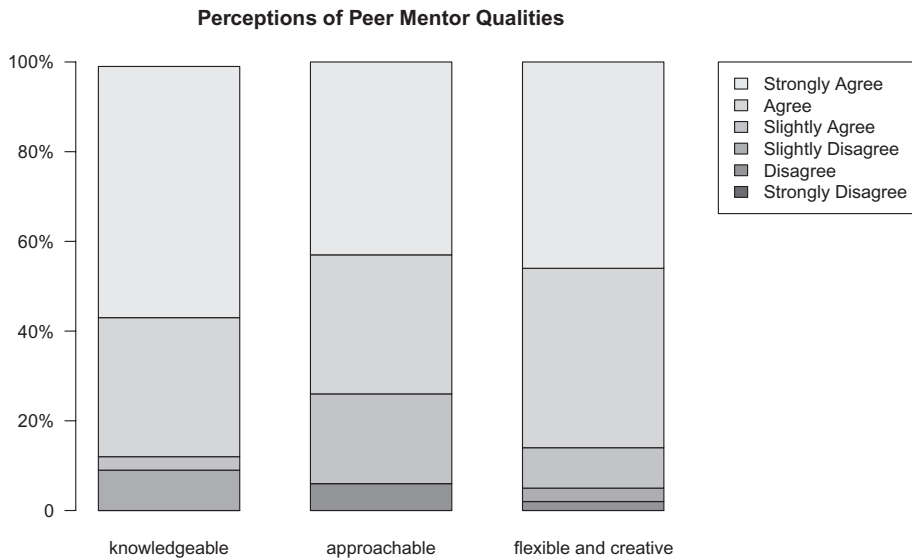


Figure 1. Student perceptions of peer mentor qualities: knowledgeable, approachable, and flexible and creative ($N = 51$).

- She's been supportive, tells me my strengths, and tells me not to give up, because she knows I've been struggling.

In addition, CS1 students credited the peer mentor sessions with contributing to their confidence and understanding, with most students strongly agreeing that the written feedback and in person meeting promoted both confidence and understanding, as shown in Figure 2. Only four students disagreed or slightly disagreed with these statements, with one noting in the optional comments that this was because her level of confidence and understanding was already high without any peer mentor intervention.

Perceptions from peer mentors: major barriers facing introductory students

By serving as peer mentors, these students contribute insight about the barriers facing students in CS1. These observations benefit the faculty instructors and provide an opportunity for reflection for the peer mentors themselves.

The peer mentors learned that time management was a major stumbling block facing CS1 students. One peer mentor noted mentees were "not realizing how long it takes to make a well written program, and then rushing in when they realize that they don't have any time left" whereas another peer mentor added that many students do not start the assignment early enough. Another peer mentor shared, "I learned that many students write comments after they complete their code, which makes their planning less effective". A third mentioned that students sometimes "rush through" without carefully reading directions.

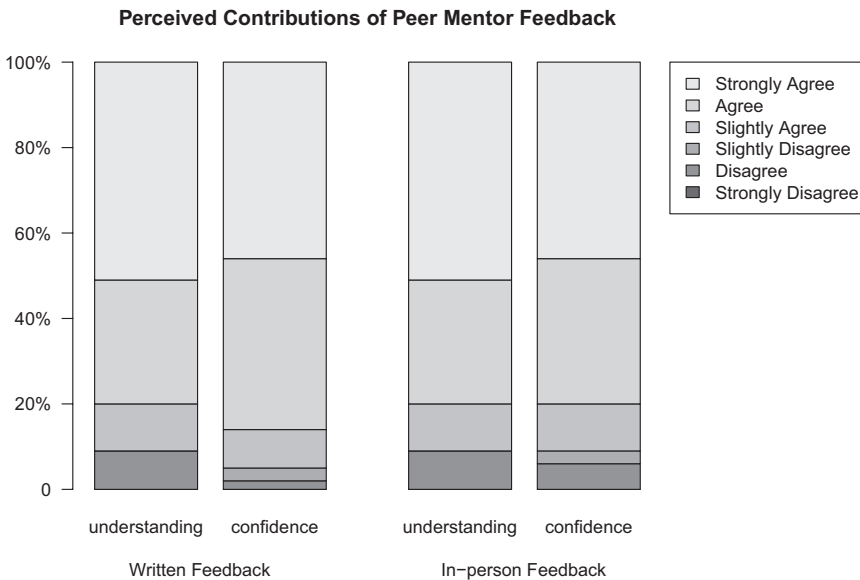


Figure 2. Student perceptions of peer mentor feedback: contributions of written feedback and in-person feedback to their own understanding and confidence ($N = 51$).

Others highlighted the importance of talking about code with introductory students. One peer mentor shared,

Often stumbling blocks are not clear from the code submitted and it is good to ask the students what they struggled with while coding or ask specifically if they struggled with when to use *if* and *if-else*, how to choose the values to loop through, etc. This helps them open up and allows clarification if necessary.

Another peer mentor noted how much coding intimidated introductory students: "I learned that computer science is scary for people. Even when they do well, they may still be scared".

Mentors also talked about the different kinds of knowledge needed to excel in computer programming. One peer mentor said that, "while students pick up the logic of variable scope and functions, deciding how to make them interact most efficiently can take more time". A second peer mentor explained, "getting used to different debugging strategies can be difficult, especially when the issue is with the design of the program or order of function calls rather than a syntax or logical error".

Advice for new students

The mentors' advice for the CS1 students also demonstrates their understanding of the coding process, with a focus both on the progressive nature of learning and strategies to help improve mastery. For example, students need to "remember how past assignments build up to the current one, and also how the current assignment help them build skills that they will use in future assignments". Another

peer mentor pointed out that students benefit from knowing, “even if they didn’t do everything perfectly, they still did well on the assignment and understood the core concepts”.

The peer mentors also emphasized their role in directing students to get help from course resources. One peer mentor suggested that students are sometimes embarrassed to ask for help, and reminds them “which resources are available when”. Peer mentors wanted to emphasize that asking can only lead to greater understanding and confidence.

Finally, peer mentors shared specific learning strategies. One peer mentor explained, “[intro students] should draw diagrams and fill out the program skeleton before trying to implement any code”. Multiple peer mentors emphasized writing comments before and during the process which helps to improve their design and will make debugging go faster. In addition, students should “look back on code they’ve already written to help with the code they are writing currently”. Going along with this, is the advice to start early and plan ahead. As one peer mentor said, and the others echoed in similar sentiments, there is value in “pseudocoding out the logic”.

Discussion

In this paper, we focused on the preparation for near-peer mentors as delivered through a training course. The near-peers offer peer code review, weekly one-on-one meetings with customized feedback, and although not the focus, they also provide active learning workshops. The preparation for this role included a careful review of the introductory computer science curriculum, as well as practice in providing written code review feedback and in-person feedback. This model emphasizes the importance of weekly feedback, focused holistically on the coding process and progress over time, from a seasoned peer. Inclusive pedagogy, including a discussion of emotional intelligence, constructive feedback, self-efficacy, and reflective practice, was an important component of the training. The results suggest that introductory computer science students perceived the near-peers as knowledgeable, approachable, and flexible or creative in their approaches to helping them to learn. They also credited the peer mentors with improvements in their confidence and understanding, whether from their code reviews or individual meetings. This suggests that the use of near-peer mentors within introductory courses, as a mechanism to provide regular feedback, can provide a valuable resource for developing the self-efficacy of introductory computer science students.

A key aspect of our model is the weekly one-on-one meetings between mentor and mentee throughout the semester. The goal is for the peer mentor to become a trusted and approachable resource for the CS1 student, enabling the student to be receptive to critical feedback, to be comfortable asking questions, and to become a self-regulated learner. While we recognize that such feedback can in

theory be provided by faculty or by teaching assistants, the time to invest in such feedback may not be possible.

The near-peer mentors provided a valuable lens into the learning of computer science students, noting the major barriers introductory students face along the way. Many students wait too long to start their programs, do not add comments as they go along, and fail to revisit past assignments to trace their own development. The near-peer mentors reveal important understandings into meta-cognitive processes (Zimmerman, 1990). As described by socio-cognitive theory, we know that self-efficacy plays an important role in taking risks in learning, including troubleshooting when one faces barriers (Pajares, 1996). Given that peer models can be effective at persuading students to persist in their learning, advice from a peer mentor – especially one trained in the ways to improve self-efficacy – to engage in the kinds of planning and recognition of progress can be especially powerful (Schunk, 1991).

Beyond the feedback, the near-peer mentors were all enthusiastic about their participation in computer science and wanted newer students to feel that same sense of mastery and satisfaction in the field. Similar to previous studies, near-peers act as ambassadors for departments in powerful ways (Bowling et al., 2015). While recruiting the first cohort of peer mentors took significant effort from faculty, by the second iteration, many students who benefited from having a peer mentor were interested in becoming peer mentors themselves. Three peer mentors from the first cohort chose to return as “lead” peer mentors in subsequent iterations. The investment in the training of near-peer mentors thus represents an investment in the department as a whole. Furthermore, it is an investment in shaping a positive climate of learning and reputation for the department.

Limitations

This project took place in one institution, a small women’s college, and thus with predominantly students who identified as women, albeit a racially, socioeconomically, and internationally diverse group of women. Given the interest in improving the recruitment and retention of women in computer science, however, we do think our work is relevant for other institutions, although may require further investigation into the composition of peer mentor cohorts when very few women are available (see Stout, Dasgupta, Hunsinger, & McManus, 2011). In addition, we only included one semester of data as an initial proof of concept. One strength, however, is we included not only peer mentor self-report data, but also the ratings of introductory computer science students. We hope the provision of the curricular design will be helpful to other institutions.

Implications for practice

We can imagine adaptations of this work in other institutions. For example, in many institutions there is a graduate assistant coordinator or a partnership with

the teaching and learning center to support the training of mentors (Streitwieser & Light, 2010). We recognize that many large research universities do not currently have a large number of women enrolled. They may look at the training program within UMBC's engineering program that includes men who are allies for women in STEM and includes coursework in gender studies for both the women and men who were enrolled (Rheingans, Brodsky, Scheibler, & Spence, 2011).

Benefits of all female STEM learning environments have been documented in summer coding programs (Lang, Fisher, Craig, & Forgasz, 2015) as well as within project teams in engineering (Dasgupta, Scircle, & Hunsinger, 2015). Computer science courses may try to experiment with grouping students in different ways to see how these principles apply in their classrooms. We also see possibilities of combining pair programming and peer code review initiatives in different ways. For example assigning one seasoned code reviewer to a course section to help with training peers to assess each other's code and to engage in better organized pair programming may be a solution that similarly improves climate.

Note

1. Referring to seasoned peers who may be just a course or two ahead of newer, introductory students. This term is sometimes used to distinguish the practice from peer mentoring where peer refers to a learner at the same stage.

Disclosure statement

No potential conflict of interest was reported by the authors.

Funding

This work was supported by the Google Foundation of Tides Foundation under a Computer Science Capacity Award.

Notes on contributors

Heather Pon-Barry is Clare Boothe Luce Assistant Professor of Computer Science at Mount Holyoke College.

Becky Wai-Ling Packard is Professor of Psychology and Education and Director of the Weissman Center for Leadership at Mount Holyoke College.

Audrey St. John is Associate Professor of Computer Science at Mount Holyoke College.

References

- Alvarado, C., Dodds, Z., & Libeskind-Hadas, R. (2012). Increasing women's participation in computing at Harvey Mudd College. *ACM Inroads*, 3, 55–64.
- Anagnos, T., Lyman-Holt, A., Marin-Artieda, C., & Momsen, E. (2014). Impact of engineering ambassador programs on student development. *Journal of STEM Education: Innovations and Research*, 15, 14–659.

- Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory*. Englewood Cliffs, NJ: Prentice Hall.
- Beyer, S. (2008). Gender differences and intra-gender differences amongst management information systems students. *Journal of Information Systems Education*, 19, 301–310.
- Beyer, S. (2014). Why are women underrepresented in computer science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education*, 24, 153–192.
- Blanc, R. A., DeBuhr, L. E., & Martin, D. C. (1983). Breaking the attrition cycle: The effects of supplemental instruction on undergraduate performance and attrition. *The Journal of Higher Education*, 54, 80–90.
- Bowling, B., Doyle, M., Taylor, J., & Antes, A. (2015). Professionalizing the role of peer leaders in STEM. *Journal of STEM Education*, 16, 30–39.
- Brought, G., Wahls, T., & Eby, L. M. (2011). The case for pair programming in the computer science classroom. *ACM Transactions on Computing Education*, 11. Article 2.
- Burke Johnson, R. (1997). Examining the validity structure of qualitative research. *Education*, 118, 282–292.
- Business Insider. (2014). Retrieved from <http://www.businessinsider.com/twitter-workplace-diversity-numbers-2014-7>
- Cheryan, S., Drury, B. J., & Vichayapai, M. (2013). Enduring influence of stereotypical computer science role models on women's academic aspirations. *Psychology of Women Quarterly*, 37, 72–79.
- Cheryan, S., & Plaut, V. C. (2010). Explaining underrepresentation: A theory of precluded interest. *Sex Roles*, 63, 475–488.
- Chisholm, R. M. (1991). Introducing students to peer-review of writing. *Writing Across the Curriculum*, 3, 4–19.
- Cohen, G. L., Steele, C. M., & Ross, L. D. (1999). The mentor's dilemma: Providing critical feedback across the racial divide. *Personality and Social Psychology Bulletin*, 25, 1302–1318.
- Crespo, R. M., Pardo, A., & Kloos, C. D. (2004). An adaptive strategy for peer review. *ASEE/IEEE Frontiers in Education Conference*. Savannah, GA.
- Dasgupta, N., Scircle, M. M., & Hunsinger, M. (2015). Female peers in small work groups enhance women's motivation, verbal participation, and career aspirations in engineering. *Proceedings of the National Academy of Sciences*, 112, 4988–4993.
- DeChenne, S. E., Enochs, L. G., & Needham, M. (2012). Science, technology, engineering, and mathematics graduate teaching assistants teaching self-efficacy. *Journal of the Scholarship of Teaching and Learning*, 12, 102–123.
- Demetriadis, S., Egerter, T., Hanisch, F., & Fischer, F. (2011). Peer review-based scripted collaboration to support domain-specific and domain-general knowledge acquisition in computer science. *Computer Science Education*, 21, 29–56.
- Garcia, D. D., Chapman, G., Hazzan, O., Johnson, M., & Sudol, L. A. (2010). Rediscovering the passion, beauty, joy and awe: Making computing fun again, part 3. In *Proceedings of the 41st ACM technical symposium on Computer science education (SIGCSE)* (pp. 394–395).
- Garousi, V. (2010). Applying peer reviews in software engineering education: An experiment and lessons learned. *IEEE Transactions on Education*, 53, 182–193.
- Griswold, A. (2010, Fall). An elemental education. *University of Maryland-Baltimore County Magazine*. Retrieved from <http://www.pkallsc.org/assets/files/UniversityofMarylandBaltimoreCounty-ChemistryDiscoveryCenter.pdf>
- Hoda, R., & Andreae, P. (2014). It's not them, it's us! Why computer science fails to impress many first years. *Proceedings of the Sixteenth Australasian Computing Education Conference (ACE2014)* (Vol. 148, pp. 159–162). Auckland.

- Hundhausen, C. D., Agrawal, A., & Agarwal, P. (2013). Talking about code: Integrating pedagogical code reviews into early computing courses. *ACM Transactions on Computing Education*, 13. Article 14.
- Kinnunen, P., & Simon, B. (2012). My program is ok – Am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education*, 22(1), 1–28.
- Lang, C., Fisher, J., Craig, A., & Forgasz, H. (2015). Outreach programmes to attract girls into computing: How the best laid plans can sometimes fail. *Computer Science Education*, 25, 257–275.
- Lave, J., & Wenger, E. (1991). *Situated learning*. Cambridge: Cambridge University Press.
- Lewis, S. E. (2011). Retention and reform: An evaluation of peer-led team learning. *Journal of Chemical Education*, 88, 703–707.
- Liebenberg, J., Mentz, E., & Breed, B. (2012). Pair programming and secondary school girls' enjoyment of programming and the subject Information Technology (IT). *Computer Science Education*, 22, 219–236.
- Liou-Mark, J., Dreyfuss, A. E., & Younge, L. (2010). Peer assisted learning workshops in precalculus: An approach to increasing student success. *Mathematics and Computer Education*, 44, 249–260.
- Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. Cambridge, MA: MIT Press.
- Mervis, J. (2010). Better intro courses seen as key to reducing attrition of STEM majors. *Science*, 330, 306.
- National Science Board. (2014). *National Science and Engineering Indicators*. Retrieved April 10, 2016, from: <http://www.nsf.gov/statistics/seind14/content/etc/nsb1401.pdf>
- National Science Foundation, National Center for Science and Engineering Statistics. (2013). *Women, minorities, and persons with disabilities in science and engineering: 2013*. Special Report NSF 13-304. Arlington, VA. Retrieved from <http://www.nsf.gov/statistics/wmpd/>
- NCWIT. (2013). Retrieved from https://www.ncwit.org/sites/default/files/file_type/aaonepager_08282013.pdf
- Packard, B. W., Marciano, V., Payne, J. M., Bledzki, L. A., & Woodard, C. T. (2014). Negotiating peer mentoring roles in undergraduate research lab settings. *Mentoring & Tutoring: Partnership in Learning*, 22, 433–445.
- Pajares, F. (1996). Self-efficacy beliefs in academic settings. *Review of Educational Research*, 66, 543–578.
- Peterfreund, A. R., Rath, K. A., Xenos, S. P., & Bayliss, F. (2008). The impact of supplemental instruction on students in stem courses: Results from San Francisco State University. *Journal of College Student Retention: Research, Theory & Practice*, 9, 487–503.
- Porter, L., Bailey Lee, C., & Simon, B. (2013). Halving fail rates using peer instruction: A study of four computer science courses. In *Proceedings of the 44th ACM Technical Symposium on Computer science education (SIGCSE)* (pp. 177–182). New York.
- Rath, K. A., Peterfreund, A. R., Xenos, S. P., Bayliss, F., & Carnal, N. (2007). Supplemental Instruction in Introductory Biology I: Enhancing the Performance and Retention of Underrepresented Minority Students. *Cell Biology Education*, 6, 203–216.
- Rattan, A., Good, C., & Dweck, C. S. (2012). "It's ok – Not everyone can be good at math": Instructors with an entity theory comfort (and demotivate) students. *Journal of Experimental Social Psychology*, 48, 731–737.
- Reardon, S., & Tangney, B. (2014). Smartphones, studio-based learning, and scaffolding: Helping novices learn to program. *ACM Transactions on Computing Education*, 14. Article 23.
- Reges, S. (2003, February). Using undergraduates as teaching assistants at a state university. *ACM SIGCSE Bulletin*, 35, 103–107.
- Rheingans, P., Brodsky, A., Scheibler, J., & Spence, A. (2011). The role of majority groups in diversity programs. *ACM Transactions on Computing Education*, 11, 1–15.

- Roberts, E. (2016). A history of capacity challenges in computer science. Retrieved June 1, 2016, from <http://cs.stanford.edu/people/eroberts/CSCCapacity.pdf>
- Roberts, E., Lilly, J., & Rollins, B. (1995, March). Using undergraduates as teaching assistants in introductory programming courses: An update on the Stanford experience. *ACM SIGCSE Bulletin*, 27, 48–52.
- Schunk, D. H. (1991). Self-efficacy and academic motivation. *Educational Psychologist*, 26, 207–231.
- Shute, V. J. (2008). Focus on formative feedback. *Review of Educational Research*, 78, 153–189.
- Søndergaard, H., & Mulder, R. A. (2012). Collaborative learning through formative peer review: Pedagogy, programs and potential. *Computer Science Education*, 22, 343–367.
- Stout, J. G., Dasgupta, N., Hunsinger, M., & McManus, M. A. (2011). STEMing the tide: Using ingroup experts to inoculate women's self-concept in science, technology, engineering and mathematics (STEM). *Journal of Personality and Social Psychology*, 100, 255–270.
- Streitwieser, B., & Light, G. (2010). When undergraduates teach undergraduates: Conceptions of and approaches to teaching in a peer led team learning intervention in the STEM disciplines: Results of a two year study. *International Journal of Teaching and Learning in Higher Education*, 22, 346–356.
- Tenenbaum, L. S., Anderson, M. K., Jett, M., & Yourick, D. L. (2014). An innovative near-peer mentoring model for undergraduate and secondary students: STEM focus. *Innovative Higher Education*, 39, 375–385.
- Tien, L. T., Roth, V., & Kampmeier, J. A. (2004). A course to prepare peer leaders to implement a student-assisted learning method. *Journal of Chemical Education*, 81, 1313–1321.
- Treisman, U. (1992). Studying students studying calculus: A look at the lives of minority mathematics students in college. *The College Mathematics Journal*, 23, 362–372.
- Tripp, T., & Rich, P. (2012). Using video to analyze one's own teaching. *British Journal of Educational Technology*, 43, 678–704.
- Wang, Y., Li, H., Feng, Y., Jiang, Y., & Liu, Y. (2012). Assessment of programming language learning based on peer code review model: Implementation and experience report. *Computers & Education*, 59, 412–422.
- Watson, C., & Li, F. W. B. (2014). Failure rates in introductory programming revisited. *Proceedings of the 2014 conference on Innovation & Technology in Computer Science Education (ITiCSE)*, 39–44. New York: ACM.
- Whitney, T., Gammal, D., Gee, B., Mahoney, J., & Simard, C. (2013). Priming the pipeline: Addressing gender-based barriers in computing. *Computer*, 46, 30–36.
- Yu, T. C., Wilson, N. C., Singh, P. P., Lemanu, D. P., Hawken, S. J., & Hill, A. G. (2011). Medical students-as-teachers: A systematic review of peer-assisted teaching during medical school. *Advances in Medical Education and Practice*, 2, 157.
- Zarb, M., & Hughes, J. (2015). Breaking the communication barrier: Guidelines to aid communication within pair programming. *Computer Science Education*, 25, 120–151.
- Zimmerman, B. J. (1990). Self-regulated learning and academic achievement: An overview. *Educational Psychologist*, 25, 3–17.


Appendix 1

Peer mentors performed written code reviews using the Upsource (by JetBrains) code review tool. The feedback comments from peer mentors appear in-line with a shaded yellow shaded background. Peer mentors tailored their feedback to the level appropriate for the mentee; e.g. feedback might be about a logical bug (Figure A.1) or it might about programming style (Figure A.2).

```

91     /**
92     * Insert a new node with data at the end of the list.
93     */
94     public void insertLast(T data) {
95         LinkedListNode<T> lastNode = new LinkedListNode<T>(); // create a new
96                                                         // node.
97         // set data.
98         lastNode.setData(data);
99
100        // set head as the current node.
101        LinkedListNode<T> currentNode = head;

```

 on Nov 13, 2016

If the linked list is empty then currentNode would be null. When you call currentNode.getNext(), you would run into Null Pointer Exception. Therefore, consider using an if statement to check and handle the case when list is empty

✓ Resolve ↩ Reply ⚙ Labels

```

102        // if empty then set the last node to the head node.
103        // it is similar to the insertLastNode function.
104        while (currentNode.getNext() != null) {
105            currentNode = currentNode.getNext();
106        }
107
108        // make the next node the last node by setting it.
109        currentNode.setNext(lastNode);
110
111    }


```

Figure A.1. Example peer code review comment about a logical bug.

```

28         for (int i=0; i < array.length - 1; i++)
29         {
30             //temp variable set to i value
31             int temp = i;

```

 on Nov 13, 2016

While temp certainly tells us that it's a temporary value, you might want to choose a more descriptive name.

✓ Resolve ↩ Reply ⚙ Labels

Figure A.2 Example peer code review comment about programming style.

Appendix 2

Questions for peer mentors

Using the 1–6 scale below, please rate the following statements:

(1 = Strongly Disagree, 6 = Strongly Agree)

- (1) I am confident in my ability to:
 - (a) Promote student participation
 - (b) Make students aware that I have a personal investment in them and in their learning
 - (c) Create a positive classroom climate for learning
 - (d) Encourage my students to ask questions during class
 - (e) Provide support/encouragement to students who are having difficult learning
 - (f) Show my students respect through my actions
 - (g) Provide detailed feedback about their academic progress
- (2) I was knowledgeable
- (3) I was helpful
- (4) I was flexible and creative in finding ways to help students
- (5) I was able to contribute to confidence-building among students I worked with
- (6) I was able to contribute to the learning of the students I assisted

What did you learn about the most common stumbling blocks facing students?

What was the most helpful piece of advice you were able to give students or the most common form of strategy you offered to students?

Questions for CS1 students

(Note: GEM, or Giga Education Mentor, refers to the peer mentor)

Using the 1–6 scale below, please rate the following statements:

(1 = Strongly Disagree, 6 = Strongly Agree)

- (1) The GEM who led my session was knowledgeable
- (2) The GEM who led my session was approachable
- (3) The GEM who led my session was flexible and creative in finding ways to help students
- (4) In the written feedback provided by your GEM:
 - (a) My GEM contributed to my confidence with the course material
 - (b) My GEM contributed to my learning of the course material
- (5) In the weekly in-person meetings with your GEM:
 - (a) My GEM contributed to my confidence with the course material
 - (b) My GEM contributed to my learning of the course material