As a teacher, my primary job is not to impart facts, but to create welcoming communities of learning in which students can have transformative encounters with beautiful ideas. I am extraordinarily committed to teaching; teaching and research both bring me great joy as inseparable halves of a unified academic pursuit.

A welcoming community of learning is a social space where students feel acceptance and true belonging in a group of people passionate about learning rather than promoting themselves. The sad fact in computer science is that many students—particularly, it seems, women and minorities—never feel this sense of belonging. Teachers have a critical role to play, in shaping the culture and social spaces in which students learn. I strive to make my courses into welcoming communities, places where every student feels they can and do belong, and indeed that there is something to belong to. In a course I designed and taught at the University of Pennsylvania, “The Art of Recursion”, I tried to foster this kind of community in several ways. First, I used the unusual format of alternating lectures and student-led discussions. Every week we devoted an entire class period to having randomly chosen students present their solutions to homework problems and then lead the class in discussing them. I tried to stay out of the way as much as possible and let the students hash out solutions on their own. Of course, with no other structure, this format has the potential to create anxiety and feelings of being judged, just the opposite of a welcoming community. The difference is that we spent time at the beginning of the semester explicitly discussing the sort of community we wanted to create, and how to go about it: to value one another as human beings; to make a clear separation between that value and the mistakes and failures inherent in our pursuit of truth; and to contribute in ways designed to promote others instead of oneself. I also tried hard to model all of this for them. In the end, it resulted in an amazing amount of respectful interaction, discussion, and collaboration. From my perspective, it seemed that the students—including several women—felt a strong sense of shared accomplishment and belonging.

At Williams, where I have been co-teaching an introductory programming course, I have seen all too clearly the ways in which some students feel that they are not good enough, that they do not belong, that they will be found out as an imposter. I try to make clear in my interactions with students that they can and do belong and that I value them and their uniquenesses more highly than I value their performance in class. I have also begun thinking of creative ways to foster a positive sense of belonging in the second-semester CS course which I will teach in the spring, including doing some variant on the celebrated values exercise [1] on the first day of class.

Once an inclusive community of learning is created, it would be a waste to simply impart facts; indeed, students to whom facts are merely imparted will never feel a part of a community in the first place. Marvelling at beautiful ideas is one my biggest personal motivations in both research and teaching, and my goal is to afford students the opportunity to have transformative encounters with beautiful ideas. I want students to come away not just as better critical thinkers and problem solvers, but as better problem posers, and as better connoisseurs of problems and solutions. The ability to pose good questions, and to appreciate good solutions, gets at the heart of the creative and scientific processes. I use a number of techniques to achieve this. First, I try to illustrate topics in fun and creative ways, not necessarily to entertain students,
but to help them get beyond their preconceived ideas and encounter computer science and mathematics in new ways. I’ve taught middle school students about finite automata using magic markers and giant sheets of paper on the floor; I’ve taught recursion using a bowl full of marbles and a ball of twine; I’ve had a confederate “interrupt” class with an inane request that turned out to illustrate a key concept. I also typically spend a good deal of class time talking about intuition and the big picture, trying to help students see where the current topic sits within a broader context and how it connects to others. Finally, I am a big proponent of “inquiry-based learning” approaches, where students are the primary drivers of the learning process. Students remember ideas they discover for themselves, because they understand the motivation and how the idea fits into a larger context.

Finally, I am extraordinarily committed to teaching and to continually improving as a teacher. During my time as a graduate student at the University of Pennsylvania, I not only served as a teaching assistant for three semesters (only two are required), but also designed and taught two of my own courses from scratch, and completed a teaching certificate through Penn’s Center for Teaching and Learning. As a TA, I chose courses where I would have real teaching duties rather than simply grading, contributing particularly to the design of Benjamin Pierce’s experimental “Software Foundations” course. I was awarded a Teaching Practicum Award, given every year to several PhD students in the CS department in recognition of their outstanding effort and enthusiasm as a TA. I went on to teach my own undergraduate course, a half-credit, semester-long introduction to the Haskell programming language, offering it a total of three times. The materials I developed for that course are now some of the materials most highly recommended by the Haskell community to people wishing to learn Haskell. As described previously, I also created an experimental full-credit course entitled “The Art of Recursion”, exploring the theory and practice of recursion from a number of viewpoints. My efforts were recognized with the Penn Prize for Excellence in Teaching by Graduate Students, a prestigious, student-nominated award given yearly to ten graduate students selected from across all of Penn’s graduate programs. Finally, I concurrently completed a teaching certificate through Penn’s Center for Teaching and Learning, which required participation in a number of workshops, a formal teaching observation, and a capstone seminar. I was also an active participant in an informal seminar for graduate students interested in CS education, where we discussed selected issues in education and gave each other feedback and encouragement in our teaching.

Courses I am willing and able to teach include introductory courses, data structures, algorithms, programming languages, theory of computation, compilers, discrete mathematics, and linear algebra. I would also be excited to offer advanced courses in functional programming or category theory.

References