Teaching Statement

Michael D. Bond

In my approach to teaching, students learn by interacting in class and collaborating outside of class. This approach is based on what has worked in my experience as a student, teacher, and mentor, particularly my experience working one-on-one with students.

Approach. To help students with diverse backgrounds, experience levels, and learning styles get the most out of classes, I will promote interactive learning in class and collaborative learning outside of class. Students will work together briefly during class to discuss problems that arise in the material and propose solutions, and they will also work together to solve brief quizzes that test understanding of a recently introduced concept. As a student, I found these exercises forced me to actually think about the material and its implications. To engage more students from diverse backgrounds, I will emphasize how computer science is relevant to areas besides computer science.

Outside of class, students will frequently collaborate, particularly when doing something new. For example, parallel programming is hard, especially when doing it for the first time, so students can benefit greatly from doing pair programming. I found pair programming to be invaluable when first learning how to use locks to protect shared resources. Students will work in groups on some assignments and projects, and they will be allowed to discuss ideas (but not actually do the work together) on individual assignments. I also plan to help organize peer tutoring, study groups, and code reviews.

Experience. In my experience, students benefit from a variety of opportunities to learn, both from instructors and fellow students. Students need material explained in different ways, they need to work together to learn difficult material, and in some cases they need new challenges to stay engaged.

As a teacher and advisor, I will draw on experience from co-supervising three graduate students and one undergraduate student. The main challenges for me have been providing guidance on short-term goals and tasks, while maintaining and promoting a long-term vision. When a student gets stuck developing software, we often look at the code together, and the student explains a section of it in detail. This usually leads to an in-depth discussion and helps me ask questions that lead the student toward a solution such as a bug fix or a way to implement new functionality. We also discuss how current low-level problems relate to long-term goals.

While a master’s student at the University of Illinois, I was a teaching assistant for Jeff Erickson’s algorithms class. In office hours, I helped students understand complex material, typically by explaining it in different ways and asking questions to see what they were not understanding and how to guide them in figuring it out. I learned traits from Jeff that I plan to emulate: how to be well prepared, well organized, and energetic in class; patient and positive with students during office hours; and fair and communicative about expectations as a manager of teaching assistants.

As an instructor, I will draw on experience working one-on-one with students. While in Austin, I mentored high school students for a year and a half: I taught and played chess with students through the Boys and Girls Club, and I tutored students in math through a school volunteering organization. These experiences taught me to be patient and positive, to ask questions, and to explain material in different ways until students understand.

Before college, I was an assistant teacher for a high school computer science course, where I helped students with lab programming assignments. Some students struggled, and they benefited from code reviews and having me explain the material in different ways. Other students completed the course material early and needed new challenges, so I developed and taught a C++ game library that helped these students learn object-oriented programming while writing video games during the final weeks of the course.

Topics. I am prepared to teach core classes in compilers and programming languages. With additional preparation, I could teach introductory programming courses and architecture, and potentially operating systems and security.