John Kloosterman — Teaching Statement

jkloosterman.net
jklooste@umich.edu

Computing skills are tools every student needs in today's world. Some students will use those skills in careers in software development, some will use them to analyze business cases or scientific data, and others will need them to create informed public policy about privacy and data ownership. As a computer science educator, my role is to show students how computation can help them achieve these diverse goals, to meet them at their different levels of preparation, and to help them build the knowledge and skills they need to succeed. I do this with engaging lectures built with active learning techniques, personal attention in office hours, and attention to diversity and inclusion in my teaching.

I have experience as a primary instructor for a programming and data structures course. Other courses I would be prepared to teach include introductory programming, computing for non-majors, data structures and algorithms, web programming, compilers, and computer architecture.

Engaging Students in the Classroom

In the classroom, I use active learning to engage students with the material and to give me insight into how students are learning. In the 223-student section of the programming and data structures class I taught, I paired students at the beginning of each class and had each pair work through an exercise after each topic in lecture. Examples of these exercises included determining when constructors and destructors would run in a code sample, drawing a diagram of a data structure, or writing code for part of an algorithm. While they worked, I walked around the classroom to observe their work and be available for individual questions. An example of the positive feedback I received about this technique came from a student in their course evaluation: "*Giving the students time to work together and figure out a way to code something allowed me to develop my algorithmic thinking.*"

Besides giving students immediate feedback on their learning, active learning can bridge the differences in coding experience and background knowledge that students bring to a computer science class. Active learning exercises at higher levels of Bloom's Taxonomy, such as designing a class hierarchy in a lecture on object-oriented design, allows students hearing about concepts for the first time to practice using them, while students working ahead can think about aspects of higher level design. This allows me to teach in a way that students coming in with less prior knowledge can understand without losing the interest of students ready for more depth.

I am intentional about creating opportunities for different types of assessment — one for students to get low-pressure feedback during active learning exercises and labs, and another for evaluating students using graded projects and exams. This gives students who need more help the chance to seek it out before there is a permanent impact on their course grade. In programming projects, my experience has shown me that careful, intentional design can go a long way to make students feel both motivated and included. One project I have used that resonated with students involved implementing an image processing algorithm; even students that might not be motivated to write code for its own sake enjoyed working on it. Also, it helped students learn to debug, since they could visualize the problems in the output images.

One-on-One Instruction

In office hours, I enjoy listening to students and giving personal attention to their questions and needs. This may be time working on the practical side of writing programs, such as how to design unit tests or how to form hypotheses of what is going wrong while debugging. Other students seek advice about study skills and strategies — after the midterm exam in the course I taught, one student said they excelled on the practice exams but underperformed on the actual exam because they felt overwhelmed. We talked through strategies such as budgeting time up front for each question.

Part of my role helping students succeed is being sensitive to the non-technical factors that impede their success. In the CS2 course I taught, I worked alongside the other instructors to introduce a lecture on impostor syndrome into the curriculum. Students have impostor syndrome when they do not own their accomplishments and feel as if they do not deserve to belong with their peers; it is especially common among women and minorities. The students responded positively — a number of them came to office hours to share how acknowledging it in class made them feel they were not alone.

Diversity and Inclusion

Because diversity is something that computer science struggles with both in academia and industry, I must be intentional about creating an environment where all students feel welcome. In a very explicit way, I had to address this during lecture when racist e-mails were sent to all the computer science students at the University of Michigan. However, subtler cultural narratives also exist that say only certain people can be computer scientists; one popular software development blogger claims that "most people seem to be born without the part of the brain that understands pointers." I will work to not only counter these negative messages, but model an inclusive attitude for my students toward who and what demonstrates excellence in computer science. One small example is highlighting in class the contributions that women like Barbara Liskov and Frances Allen have made to the concepts and tools that students use every day.

Professional Development

As my career in teaching progresses, I am committed to continually improving my teaching. One part of this will be gathering formative feedback. A facilitator I invited to a class session observed my teaching and collected feedback from students. Many students said they wanted to see more types of content in the lecture, which I addressed by including more live demonstrations of topics like debugging and working through longer exam-style exercises in class. Other components of my professional development will include continuing my involvement with the computer science education community in SIGCSE, as well as engineering my own programming projects to stay current with new languages and frameworks.

Conclusion

As computing plays a larger and larger role in how the world works, my teaching gives students the skills to participate in that world, whatever their background may be. In lecture, active learning gives students immediate feedback on their understanding. In office hours, I am there to listen to students and work with them on practical coding and study skills. With inclusion as a goal in my teaching and curriculum, I will work to make all students feel they belong in the computer science community, and I am committed to developing my teaching skills throughout my career.