My teaching and mentorship across the humanities and computer science are anchored in a commitment to **interdisciplinarity**. I aspire to take the best from different disciplines to encourage a computing practice that refuses to disavow its impacts through an apolitical stance, and to cultivate a culture of open curiosity, collaboration, and group work in service of communicating a vision of greater computing freedom for students across academic departments. As a Ph.D. student at Brown University, I designed and taught two courses as the Instructor of Record in the departments of Computer Science and Modern Culture and Media. Both courses presented students with new perspectives on computing freedom from 'within' one of these disciplinary frameworks by revealing the value of ideas occurring outside the expected departmental contours.

I firmly believe that interdisciplinary pedagogy yields equitable and durable learning outcomes when disciplines come together from a place of genuine curiosity, mutual respect, and a willingness to learn unfamiliar vocabularies and ways of working. My pedagogy takes its cue from the idea that it is a persevering proximity between computer science and humanistic methods that leads to the most serious and valuable reflection on the relative merits of each. This approach suggests to students that interdisciplinarity does not dilute rigor, but rather reinforces and refines the quality of the questions we ask. My teaching philosophy is thus dually defined by 1) **practice-based pedagogy** in computer science and 2) **writing as method** in the humanities.

Practice-based pedagogy in computer science

In my experience, CS students are most clearly able to see the importance of critical thinking when it directly applies to the coding problems they are tasked with solving. This 'hands-on' form of questioning can be more effective than open-ended discussion questions regarding the ethics of technology—though these too have their place. As the Graduate Advisor to the Socially Responsible Computing program in the Brown Department of Computer Science, I observed firsthand the challenges and immense value of thinking critically about how best to teach computer science to undergraduates. Some changes that we discussed (and, when appropriate, implemented) were tangible, such as modifying a distributed systems offering so that the course project was not a digital bank that exacts penalties upon wayward depositors, but an application that distributes credit 'fairly' to subscribing users on the basis of their need. Other changes amounted to structural critiques that forced us ask: what, in its essence, *is* computer science in its essence: an academic science of mathematical and machinic optimization, an engineering or 'trade' discipline that steadies software developers for the Big Tech market, or something else entirely?

Inspired by these questions, in 2022–23 I facilitated, co-designed, and taught two iterations of a practice-based course in computing freedom titled *CS for Social Change* that gives students the experience of building software for a real-world nonprofit organization. Students are partnered in groups of about five and assigned to work with organizations such as the Student Clinic for Immigrant Justice, the Social Media Analysis Toolkit (now OpenMeasures), Forensic Architecture, and the Center for Health and Justice Transformation. Organizations were selected on the basis of having real technical issues that were reasonably scoped such that a group of CS undergraduates could make meaningful progress on them over the course of one semester.

CS for Social Change is thus a course that aims to open up alternative ways of thinking about how programming can positively impact social change. The course's heavy oversubscription—some 40 students applied for one of the limited 20 slots on both occasions that I ran it—made clear to me the value of nontraditional approaches to software pedagogy. Many students expressed to me during my office hours that, although their CS education opened standard pathways to working in industry at startups and tech firms, it was not always obvious how they could apply the skills learned in their degrees to directly assist social justice work, nonprofit organizations, or other initiatives driven principally by social rather than financial imperatives.

In addition to group work with partnering organizations that I advised as the course instructor, I led a weekly seminar to reflect with reading assignments on 1) why and when we should write software to solve problems (does it do more harm than good?), 2) what we should consider when designing software (user skill set, maintenance, privacy concerns), and 3) what social, institutional, and

political structures influence software's impact on the world (capitalism, the constraints of the nonprofit sector, resource availability). One of the course's learning goals was to deconstruct the impulse to always build *new* software, an impulse that is very pronounced in undergraduate CS majors at Brown, particularly the very successful and talented senior students who made up the majority of my course. (Of the 21 students in the 2023 iteration, roughly half were final-year seniors, and almost all had industry internship experience.) In many cases, projects with partnering organizations were at first imagined as a custom web application using Django, Node, or a similar framework. By the end of the course, after considering factors such as how the nonprofit organization would maintain and update such an application, many projects were rescoped to an adaptation or enhancement of existing software used by the nonprofit, such as a Google Sheets plugin or an illustrative Jupyter notebook.

CS for Social Change encapsulates my teaching philosophy of learning the value of critical thinking through the practice of writing real software, an approach I call **practice-based pedagogy**. I am motivated to teach courses that allow students to develop their coding skills while simultaneously reflecting on when, how, and why code *should* be written.

Writing as method in the humanities

In 2024, I offered an undergraduate seminar in the Department of Modern Culture and Media titled Capitalism and Computers in the Era of AI consisting of thirteen students from both the humanities and the sciences. Like *CS for Social Change*, I designed this course with interdisciplinarity and persevering proximity in mind. The course aims to show the *practical* value of interdisciplinarity by revealing that the 'technical' history of computer science can frame philosophical questions in critical theory, such as the present-day durability of capitalism, in a new light.

If *CS for Social Change* is a course that reflects critically on writing *code*, this course is focused on writing as a critical method for thought. During class seminars, I focused on clearly breaking down the arguments in works of critical theory to ensure that students from different backgrounds could see their main points even when the genre or vocabulary felt unfamiliar. I was therefore proud to receive written anonymous feedback from students such as: "Lachlan is great at tailoring his conceptual explanations to his audience, making it much easier to engage with concepts and ideas like Neoliberalism or Marxism", and "Lachlan's a phenomenal instructor and communicator. He does a great job at unpacking the readings in-class & communicating these tricky concepts".

I seek to make critical theory relevant to students by showing its practical value. As LLM interfaces such as ChatGPT were on the rise at the time (in September 2024), I allowed students to write a weekly summary of assigned texts with the help of an LLM, asking that they also add a note on method indicating *how* exactly they prompted the LLM to produce the summary. By way of demonstration, I first showed students two summaries—one written by me, and the other by an LLM—of Turing's paper *Computing Machinery and Intelligence*. (These examples are available on the course website and included in the dossier.) This exercise is intended to encourage students to think critically not only about whether LLMs are capable of 'doing their homework', but also whether there is *value* in deferring such tasks to an LLM. Reading and writing, this pedagogy aims to suggest, are not instrumental activities as fear-mongering suggestions that AI will soon replace teachers—and readers—assume them to be, but inherently valuable ones that we should pursue for their own sake.

In addition to these two courses, I have conceptualized and helped implement changes in courses across the CS curriculum at Brown, from Introductory Programming to Cryptocurrency, through my role in the Socially Responsible Computing program. I have been a Graduate TA for Database Management Systems in CS, Head TA for *Theories of Modern Culture and Media*, and TA for *Digital Media* in the Department of Modern Culture and Media. I have also mentored and written letters of recommendation for students at Brown University, Goldsmiths, and the University of Auckland. I would be excited to teach both introductory and advanced courses in software development, media and technology studies, computing ethics, and philosophy and critical theory to work towards a more concrete pedagogy of computing freedom in the academy.