Dylan Zaragoza

Option 1: Computer Science Curriculum

1. **What are the strengths and weaknesses of the CSE program?**

   Just like any other school, earning a Computer Science degree at the University of Notre Dame has its own strengths and weaknesses. As freshmen (and freshwomen), the only real college specific course engineering students are required to take is Introduction to Engineering (I in the Fall and II in the Spring). Looking back at it now (without rose-tinted glasses), this first-year course diversity can be as much helpful as it is harmful. On one hand, the atmosphere is very welcoming and doesn't pressure students into any particular major. Its flexibility allows them to fulfill general University requirements up front, and even budgets room for some inter-college mobility going into a student's second or third year. The drawback, however, is slower-than-average pacing when compared to other universities of the same level. Some students, such as myself, who applied to Notre Dame with full intentions of majoring in Computer Science may have even felt like they did not learn anything new (other than exposure to MATLAB) their freshman year. On the other end of the spectrum, there is some controversial discussion about the order of some junior and senior level required courses and whether or not it is placing Notre Dame students at a disadvantage when it comes time for job interviews.

2. **What courses would you add, remove, modify?**

   While Sophomore year introduces C and C++ programming in Fundamentals of Computing (as well as Discrete Math, Logic Design, and just recently, Unix), I cannot help but feel as though the curriculum does not give freshmen (and freshwomen) enough credit to be able to grasp some of these concepts a year earlier. It may just be because I had some experience programming in high school, but I earnestly believe students of Notre Dame caliber can begin learning C and C++ (or even Unix) during their freshman year if given the opportunity and reason to do so. Removing Introduction to Engineering and shifting some of these topics down to freshman year would also allow this faster pace to trickle down from higher level classes. Data Structures could move from junior to sophomore year and Algorithms from senior year to junior year. This would also address the issue of covering both Data Structures and Algorithms in time for upperclassman internships and interviews.

3. **What skills or technologies need more coverage?**

   There are many courses I took as technical electives that I feel should at least be common knowledge for every computer scientist. Networking, security, and cloud computing may have more to do with networking than programming itself, but in the twentieth century I feel one would be hard pressed to find a computer-related job that has absolutely nothing to do with networking (or clouds). Learning programming in a vacuum without any exposure to networking would be like learning about the parts of a car and how to drive, but not the rules of the road or how to read signs. To tell the truth though, I'm finding it difficult to place these courses in the Notre Dame CS curriculum without replacing either 1) Philosophy or Theology (not likely since these are University requirements), 2) Technical Electives or 3) Free Electives. I'm not sure replacing the freedom electives give would be well received either, but at the very least I feel some exposure to networking should be mandatory.

4. **What are some projects or assignments that students should experience?**

   I'm not sure I would significantly change the final project for any particular course. Most projects incorporate elements of real world computer science in their design, including working in a team and seeing a product through from concept to implementation to presentation. Some students are skeptical of the learning value of the open source final project for Data Structures, but I believe it represents another important aspect of the Computer Science profession: a lot of the time we are called upon to build on top of (or fix) already existing systems. Having to wade through another programmer's code (and sometimes without documentation or reference) is less a possible situation than an eventual one, so contributing to an open source project gives students some much (under)appreciated exposure. Admittedly, the rubric for the project should probably be changed a bit to more closely reflect the topic of the class, but I stand by my opinion that it should be kept a part of the undergraduate experience.

5. **What can faculty and staff do to improve the program?**

   My undergraduate experience as an engineer at Notre Dame was all I could have hoped for. As far as I am concerned, the faculty and staff of the Computer Science program did their job of kick starting a lifelong programmer. To be perfectly honest, even while singing praises of the staff I cannot think of a single improvement to suggest (other than the curriculum changes mentioned above). The sense of faith and camaraderie I've gained here will stay with me for many years to come, both as a programmer and as a person. And I wouldn't replace that feeling with anything.