Teaching Statement

Mukesh Tiwari

My teaching philosophy is not to immediately reach a solution but to develop a thinking process (problem solving mindset) that leads to the solution. I believe every student is different and has a unique style of learning, and my role is to help them find and hone their style. When I started teaching as an assistant professor at the International Institute of Information Technology (IIIT), Bhubanesware, India[1], my single biggest challenge was keeping the students engaged in my class, especially the first year students in C programming course. At the IIIT, I taught C programming to first year students, Compiler Design and Java programming to third year students, and Cryptography to final year (4th year) students. In each course, every single problem, more or less, boiled down to keeping the students engaged in a topic. In order to keep them engaged in a class, I took a Coursera course on learning[2] and read many academic articles and non-academic articles about effective learning. I tried some of the techniques suggested in the Coursera course and academic and non-academic articles in my classes. Below I describe my experiences with teaching and efforts to engage my students.

1 Setting Clear Goals

In every course, I started with the end goal of the course. For example, in C programming course, I told my students that by the end of this course they would be able to write simple C programs, e.g., calculator, validating a debit card, and other simple real-world problems. The rationale was to show a vision to excite them for learning and instill the feeling of empowerment, from being a consumer to being a developer of software programs.

2 Start with Why

One thing that I learnt by teaching for 3 years is that if you want a concept to stick in someone's mind, then start with a *why*[3]. For example, when I introduced functions in C programming course, I wanted to convey the need of functions. Therefore, I started the class by asking them to write a program, using pen and paper, of **factorial of a number** n ($n!$). Every one was comfortable with loops, so it was quick. I then asked them to write another program: k^{th} **power of the factorial of** n ($(n!)^k$). In this assignment, some added an outer loop to cover the factorial computation, and some added another loop after the factorial computation (stored the factorial computation result and used it in later computation). At this point, I asked the class how one could write a program of **factorial of the** k^{th} **power of the factorial of** n ($((n!)^k)!$). The class slowly realised that they needed to duplicate a lot of code. I then introduced functions and showed them the solution of the previous problem by function composition.

[1] It is a small technical school

[2] `https://www.coursera.org/learn/learning-how-to-learn`

[3] I learnt this idea by reading the book *Start With Why* by Simon Sinek.

3 Catching my Mistakes

To promote active participation, at the beginning of every class I would announce that on a few occasions I would make deliberate mistakes in solving a problem, and the goal of the class was to catch me on the spot. If the class succeeded, they received class participation marks, otherwise I told them my mistake and no one received any marks. In every class, especially in programming, first I would teach a topic, and later I would solve some problems on a blackboard, related to the topic. It was during the problem solving where I committed deliberate mistakes. It increased the class participation by an order of magnitude[4]. In every single feedback that I got during my 3 years of teaching, almost everyone appreciated this idea of making deliberate mistakes.

4 Potential Courses at Berkeley

I feel qualified to teach most of the computer science courses because of my background in computer science, but my natural preference is the courses close to my research area, e.g., theorem proving, cryptography, logic and its applications in computer science, discrete mathematics, algorithms and data structures, introductory programming course (C/C++/Java/Python/Haskell/OCaml), foundation of computing, etc. The courses I can teach at Berkeley are: (i) CS 3. Introduction to Symbolic Programming, (ii) CS 9C. C for Programmers, (iii) CS 9D. Scheme and Functional Programming for Programmers (iv) CS 9H. Python for Programmers, (v) CS 61A. The Structure and Interpretation of Computer Programs (I would love to teach this), (vi) CS 61B. Data Structures, (vii) CS 70. Discrete Mathematics and Probability Theory, (viii) CS 161. Computer Security, (ix) CS 171. Cryptography, (x) CS 263. Design of Programming Languages.

In addition, I would also like to design a course to teach various voting methods used around the world and their advantages and disadvantages from a social choice theory perspective. Apart from these topics, I will be more than happy to teach other courses, given enough preparation time, at introductory and intermediate levels, e.g., type theory, theory of programming languages, networking, operating systems, databases, etc. Finally, I would like to emphasize the my teaching philosophy is not to immediately reach a solution but to develop a thinking process (problem solving mindset) that leads to the solution.

[4]I learnt this from advertising agencies where they write some ads wrong intentionally to grab the attention.