CS 421 - Programming Languages & Compilers

Instructors:
This course has recently been taught by Sam Kamin in the spring and Elsa Gunter in the fall. However, Professor Kamin has retired after the Spring 2013 semester, so this pattern is sure to change in the future.

Prerequisites:
CS 232 Computer Architecture II and CS 374 Intro to Algorithms are listed as the official prerequisites for this class. It is a good idea to be familiar with basic computer architecture concepts (such as machine-language instructions) and lower-level software concepts (such as the stack and heap) in order to understand the motivation behind many of the concepts discussed in the class. Additionally, experience with data structures (namely trees) and programming experience in at least one language is essential to complete the MPs; all MPs are done in OCaml, a functional language. As such, ECE 220 Computer Engineering and CS 225 Data Structures are strongly recommended before taking this class. During the Spring 2013 semester, few concepts were directly taken from CS 373, so students of sufficient mathematical maturity may be able to take both classes at the same time.

When to Take It:
As CS 421's follow-up course, CS 426 Compiler Construction, is only offered during the fall semester, it is likely best to take CS 421 during the spring semester of junior year.

Class Content:
CS 421 begins by introducing the functional language OCaml. Unlike languages like C, the body of a function in OCaml is simply its return value; OCaml also provides for powerful functionality such as pattern-matching, allowing programmers to express more complex concepts in fewer lines of code. The class then discusses lexing and parsing, two steps needed to transform the plain text of a source file into an Abstract Syntax Tree (AST), a compiler-internal representation of a program. Topics such as interpretation, compilation, and type-checking using Structural Operational Semantics (SOS) rules are then discussed. The class also briefly covers topics such as garbage collection and lambda calculus.

Work:
In Spring 2013, the class was broken up into 12 different assignments, 10 of which were MPs in OCaml and 2 of which were written assignments. Each assignment was relatively short, often requiring less than 100 lines of OCaml code, and took from 3-8 hours to complete. Assignments were assigned roughly weekly. The class also had two 75-minute midterm exams and one final, which was cumulative in the Spring 2013 semester.

Life After:
This class is a useful look into salient compiler concepts and will likely expose students to useful programming techniques; while many students may not need to implement full compilers in the future, lexing and parsing knowledge can be applied to many situations where the given representation of program input is not convenient for internal use. This class can be followed up by CS 426 Compiler Construction, which deals with the compilation process in more depth and introduces the concept of compiler optimization.