1. **Number and title of course:** EECS 61A, The Structure and Interpretation of Computer Programs

2. **Course objectives:** We are interested in teaching you fundamental principles about programming, not details about any particular programming language. We consider a series of techniques for controlling program complexity, such as functional programming, data abstraction, object-oriented programming, and query systems. To get past generalities you must have programming practice in some particular language, and in this course we use Scheme, a dialect of Lisp. This language is particularly well suited to the organizing ideas we want to teach. Our hope however, is that once you have learned the essence of programming, you will find that picking up a new programming language is but a few days’ work. Since many of you will have some skill programming in other languages, we hope that you will relate the concepts from this course to your previous experience.

3. **Topics covered:**
   - Functional Programming
   - Higher-Order Procedures
   - Recursion and Iteration
   - Data Parallelism using the MapReduce protocol
   - Data Abstraction
   - Hierarchical Data
   - Generic Operators
   - Object-Oriented Programming
   - Assignment, State, Environments
   - Mutable data, queues, tables, vectors
   - Concurrency, Client/Server Programming
   - Streams
   - Metacircular Evaluator, Analyzing Evaluator
   - Lazy Evaluator, Nondeterministic Evaluator
   - Logic Programming

4. **Contribution of course meeting the professional component:** (To be determined by the Undergraduate Study Committee)

5. **Relationship of course to program outcomes:** Students gain the ability to analyze and solve computer science and engineering problems through application of fundamental knowledge of mathematics, science, and engineering. Students learn to apply modern skills, techniques and tools to create a system to meet desired needs.

6. **Updated by:** Brian Harvey (5/2008)