CSE 341: Programming Languages
Course Information and Syllabus
Spring 2004

Last updated: March 24, 2004. Ignore older or undated versions.

Logistics and Contact Information: The instructor is Dan Grossman. See the course homepage (`www.cs.washington.edu/education/courses/cse341/04sp`) for information regarding teaching assistants, office hours, sections, etc. *You must join the class email list and check email at least once every 24 hours.*

Goals: Successful course participants will:

- Internalize an accurate understanding of what functional and object-oriented programs mean
- Develop the skills necessary to learn new programming languages quickly
- Master specific language concepts such that they can recognize them in strange guises
- Learn to evaluate objectively the power and elegance of programming languages and programming-language constructs
- Attain reasonable proficiency in ML and Scheme and, as a by-product, become more proficient in languages they already know

Text: The listed text for the course is: "Jeffrey D. Ullman. Elements of ML Programming, ML'97 Edition. 1998." We will not follow the text closely, but it will likely prove useful during the first few weeks. Other recommended texts, most of which are available for free, will appear on the course homepage.

Grading and Exams:

Midterm	20%	April 28 (in class)
Final	25%	June 10, 8:30–10:20
Homeworks	55%	approximately weekly

For homeworks, we may have one or two "large-homeworks/small-projects" that are worth twice as much as the other homeworks.

Do not miss the midterm or final.

Late Policy: Homework will always be due at 9:00AM on the due date. This deadline is strict. Therefore, it is exceedingly unlikely that skipping class or being late to class because of homework is in your interest. For the entire quarter, you may have three "late days". You are strongly advised to save them for emergencies. You may not use more than two for the same assignment. They must be used in 24-hour chunks.

Academic Integrity: Any attempt to misrepresent the work you did will be dealt with via the appropriate University mechanisms, and your instructor will make every attempt to ensure the harshest allowable penalty. The guidelines for this course and more information about academic integrity are in a separate document. *You are responsible for knowing the information in that document.*

Advice:

- In every course, there is a danger that you will not learn much and therefore lose the most important reason to take the course. In 341, this danger is severe because it is easy to get "distracted by unfamiliar surroundings" and never focus on the concepts you need to learn. These surroundings include new syntax, programming environments, error messages, etc. Becoming comfortable with them and appreciating their importance is *only one* aspect of this course, so you must get past it. When we move to a new language, you must spend time on your own "getting comfortable" in the new setting as quickly as possible so you do not start ignoring the course material.
- If you approach the course material by saying, "I will have fun learning to think in new ways" then you will do well. If you instead say, "I will try to fit everything I see into the way I already look at programming" then you will get frustrated.

Approximate Topic Schedule (Subject to Change):

- Syntax vs. semantics vs. idioms vs. libraries vs. tools
- ML basics (bindings, conditionals, records, functions)
- Recursive functions and recursive types
- Pattern Matching
- Higher-order functions
- Lexical vs. dynamic scoping
- Syntactic sugar
- Equivalence
- References and cycles
- Laziness and memoization
- Exceptions
- Parametric polymorphism and container types
- Abstract types and modules
- Dynamic vs. static typing
- Subtyping for records and functions
- Macros
- Closure conversion
- Garbage collection
- Abstract datatypes with dynamic typing
- Object-oriented programming is dynamic dispatch
- Pure OO
- Class-based subtyping
- Multimethods
- Relating concepts to Java
- Language-design principles
- Language-learning principles

To learn these concepts in the context of real programming languages and to gain experience with different languages, we will use the following (in order):

- Standard ML (approximately 4–5 weeks)
- Scheme (approximately 2–3 weeks)
- Smalltalk (approxmately 2 weeks)
- Java (less than 1 week)

There are thousands of languages not on this list, and many programming paradigms not represented.