CSE 341: Programming Languages
Course Information and Syllabus
Autumn 2005

Logistics and Contact Information: See the course homepage (`www.cs.washington.edu/education/courses/cse341/05au`) for information regarding teaching assistants, office hours, sections, etc. *You should 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 the power and elegance of programming languages and their constructs
- Attain reasonable proficiency in ML, Scheme, and Smalltalk
- As a by-product, become more proficient in languages they already know

Text: The "required" text 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. The "recommended" text is: "Mark Guzdial. Squeak: Object-Oriented Design with Multimedia Applications. 2001." We will cover only material corresponding to the first two chapters and online resources may suffice. You must decide how much you benefit from having a book in your hand. There is no text for the Scheme portion of the course; online resources will suffice.

Grading and Exams:

Midterm	20%	Wed, November 2 (in class)
Final	25%	Thursday, December 15, 8:30–10:20
Homeworks	55%	approximately weekly

Unless announced otherwise, all homeworks contribute equally to the 55%.
Do not miss the midterm or final.

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 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):

1. Syntax vs. semantics vs. idioms vs. libraries vs. tools
2. ML basics (bindings, conditionals, records, functions)

3. Recursive functions and recursive types
4. Datatypes, pattern matching, exceptions
5. Higher-order functions
6. Lexical vs. dynamic scope
7. Currying
8. References and cycles
9. Syntactic sugar
10. Equivalence and effects
11. Abstract types and modules
12. Parametric polymorphism and container types
13. Type inference
14. Scheme basics
15. Dynamic vs. static typing
16. Laziness and memoization
17. Implementing higher-order functions
18. Continuation-passing idioms
19. Macros
20. Abstract datatypes with dynamic typing
21. Smalltalk and Squeak basics
22. Object-oriented programming is dynamic dispatch
23. Pure object-orientation
24. Implementing dynamic dispatch
25. Subtyping for records, functions, and objects
26. Class-based subtyping
27. Fragile superclasses, multiple inheritance
28. Unexpected change via subclassing
29. Multimethods
30. Static overloading
31. Relating concepts to Java
32. Subtype vs. bounded quantification
33. Contrasting extensibility with object-orientation and datatypes
34. Basic garbage-collection implementation

To learn these concepts using real programming languages and to gain experience with different languages, we will use:

- Standard ML (a statically typed, mostly functional language) (approximately 4–5 weeks)
- Scheme (a dynamically typed, mostly functional language) (approximately 2–3 weeks)
- Smalltalk (a dynamically typed, object-oriented language) (approxmately 2 weeks)
- Java (a statically typed, object-oriented language) (less than 1 week)

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