CSE 341: Programming Languages
Course Information and Syllabus
Spring 2011

Logistics and Contact Information: The instructor is Hal Perkins. See the course homepage (`www.cs.washington.edu/education/courses/cse341/11sp`) for information regarding teaching assistants, office hours, sections, etc. *You must ensure your email settings are correct for receiving course email-list messages, which is sent to your official UW mail address, and you should check email at least daily.* We will limit email traffic to essential messages from course staff. Please use the Catalyst discussion board to stay in touch with each other outside of class.

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 Ruby
- As a by-product, become more proficient in languages they already know

Grading and Exams: Do not miss the midterm or final.

Midterm	20%	Wednesday April 27, in class (tentative)
Final	25%	Thursday, June 9, **8:30**-10:20
Homeworks	55%	approximately weekly (7 total)

Unless announced otherwise, all homeworks contribute equally to the 55%.

Late Policy: Homework is due at **11:00PM** on the due date This deadline is strict, so it is exceedingly unlikely that skipping class or arriving late to do homework is in your interest. For the entire quarter, you have **4** "late days". You are strongly advised to save them for emergencies. You may use at most 2 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.*

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" texts are: "Dave Thomas. Programming with Ruby. 2005." and "R. Kent Dybvig, The Scheme Programming Language". The course will overlap only with several chapters of these books. You must decide how much you benefit from having a book in your hands; there are also many free on-line resources.

Advice:

- Your instructor aims for lecture and section to be some of the most enriching hours of your college career. **We will start promptly, and you should arrive punctually and well-rested.**
- In every course, there is a danger that you will not learn much and thus 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, editors, error messages, etc. Becoming comfortable with them is *only one* aspect of this course, so you must get past it. When we use 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. By the end, it will relate back to what you know, but be patient.

Approximate Topic List (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. Algebraic datatypes, pattern matching
5. Higher-order functions; closures
6. Lexical scope
7. Currying
8. Syntactic sugar
9. Equivalence and effects
10. Parametric polymorphism and container types
11. Type inference
12. Abstract types and modules
13. Scheme basics
14. Dynamic vs. static typing
15. Laziness and memoization
16. Implementing higher-order functions
17. Macros
18. Abstract types via dynamic type-creation
19. Ruby basics
20. Object-oriented programming is dynamic dispatch
21. Pure object-orientation
22. Implementing dynamic dispatch
23. Subtyping for records, functions, and objects
24. Class-based subtyping
25. Multiple inheritance, interfaces, and mixins
26. Multimethods vs. static overloading
27. Subtyping vs. parametric polymorphism
28. Functional vs. OO extensibility
29. 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)
- Ruby (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, many programming paradigms not represented, and many language constructs and concepts that it would be great to study.