CSE341: Programming Languages
Course Information and Syllabus
Winter 2013

Logistics: The instructor is Dan Grossman. See the course homepage, `www.cs.washington.edu/education/courses/cse341/13wi`, for information about teaching assistants, office hours, etc. *You must ensure your email settings work for promptly receiving course email-list messages.*

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 the ML, Racket, and Ruby languages — and, as a by-product, become more proficient in languages they already know

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

Midterm	20%	Friday February 8, in class
Final	25%	Thursday March 21, 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. For the entire quarter, you have **3** "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. *Advice: Do not skip class or section to work on homework — this will cost you time in the long run.*

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.*

Texts: The instructor has developed written reading notes and videos for most of the material in the course. Therefore, the texts are optional, but they may nonetheless be very useful to some students as an alternate perspective and source for explanations. The texts are "Jeffrey D. Ullman. Elements of ML Programming, ML'97 Edition. 1998." (for the first half of the course) and "Programming Ruby 1.9, The Pragmatic Programmers' Guide (Facets of Ruby), Dave Thomas et al, 2009." (toward the end of the course). In the middle of the course, we will refer to the Racket User's Guide, which is available for free online.

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. Benefits of no mutation
5. Algebraic datatypes, pattern matching
6. Tail recursion
7. Higher-order functions; closures
8. Lexical scope
9. Currying
10. Syntactic sugar
11. Equivalence and effects
12. Parametric polymorphism and container types
13. Type inference
14. Abstract types and modules
15. Racket basics
16. Dynamic vs. static typing
17. Laziness, streams, and memoization
18. Implementing languages, especially higher-order functions
19. Macros
20. Eval
21. Abstract types via dynamic type-creation and simple contracts
22. Ruby basics
23. Object-oriented programming is dynamic dispatch
24. Pure object-orientation
25. Implementing dynamic dispatch
26. Multiple inheritance, interfaces, and mixins
27. OO vs. functional decomposition and extensibility
28. Subtyping for records, functions, and objects
29. Class-based subtyping
30. Subtyping vs. parametric polymorphism; bounded polymorphism

To learn these topics using real programming languages and well-suited for them, we will use:

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

There are thousands of languages not on this list, many programming styles not represented, and many language constructs and concepts that it would be great to study.