CSE 505: Concepts of Programming Languages
Course Information and Syllabus
Winter 2012

Logistics and Contact Information: The instructor is Dan Grossman. See the course home page (`http://www.cs.washington.edu/education/courses/cse505/12wi/`) for all pertinent information.

Goals: We will investigate concepts essential to programming languages including control flow, scope, functions, types, objects, and threads. As time permits, we will explore other concepts such as continuations, exceptions, message-passing, and connections to logic. Our primary intellectual tools will be *operational semantics* and *Caml programs*. Prior knowledge of neither is required nor expected. We will build formal models and prove properties about them, which is a useful research skill throughout computer science.

Successful course participants will learn to:

- Give precise definitions to programming-language features
- Prove properties of inductively defined sets (e.g., well-typed programs)
- Appreciate some programming-language theory jargon (e.g., inference rules) and find the research literature more approachable
- Write better programs by exploiting modern language features such as higher-order functions and objects

In short, our goal is to use "theory" to make us better programmers and better researchers. Always think, "how is this related to programs I have written?"

Audience: This course is an introductory course for CSE Ph.D. students. Others should contact the instructor to discuss appropriateness. It is intended for students who will never take another languages course and those who will pursue this area indefinitely. Experience with functional languages, proofs by induction, and logic may prove useful. (Having such experience might let you "slack" a bit; students without it should not necessarily run away.)

Format: Two weekly lectures will develop the course content. The textbook (Types and Programming Languages by Pierce) covers some of the same material. It can serve as an excellent "second explanation" but we will not follow it closely, so you may consider it optional. Homeworks will extend the material discussed in lecture; expect to learn as you do them. Programming exercises must be done in the Caml language, which will be discussed in class. Exams will assess understanding of the lectures and homeworks.

Homeworks, Exams, and Grading: We will have five homeworks, one short writing assignment, one midterm, and one final. Each homework and the writing assignment will be worth 1/9 of the course grade. Each exam will be worth 1.5/9 of the course grade. This is subject to change (unlikely).

Academic Integrity: Any attempt to misrepresent the work *you* have done will result in the maximum penalty the university allows. If there is any doubt, indicate on an assignment who assisted you and how. In general, you may discuss general approaches to solutions, but you must write your solutions on your own. You should not show your written solution to someone else or view someone else's solution. Particular assignments may include more specific instructions. If not, ask. *Violating the academic trust your instructor and classmates have placed in you will have a* ***far worse*** *effect on your academic future than doing poorly on a homework assignment.*

Advice: It is likely you can pass this course without appreciating what we are doing or retaining much of what you learned. Doing so is a poor use of your time. To "appreciate" and "retain" it is best to:

- Determine how each lecture topic and homework problem fits into the course outline and course goals.
- Master the details of the early topics in the course. The material builds on early material more than it seems.

Probable Topics: (subject to change)

- Introduction to Caml
- Inductive definitions
- Operational semantics
- "Pseudodenotational" semantics
- Assignment and basic control flow
- Scope and variable binding
- Lambda calculus
- First-class continuations
- Continuation passing style and the CPS transform
- Simple types
- Type safety
- Curry-Howard isomorphism
- Universal and existential types
- Recursive types
- Shared-memory concurrency (threads, locks, transactions, memory-consistency models)
- Message-passing concurrency; Concurrent ML
- Object-oriented programming
- Inheritance
- Multimethods

Probable Nontopics: There are many topics that are appropriate for this course but we simply will not have time for. We will try to give a hint about what we are missing with respect to the following or squeeze some in time permitting.

- Parametricity
- Type-and-effect systems
- Type inference
- Denotational semantics
- Axiomatic semantics
- Module systems
- Monads/workflows
- Programming with lazy evaluation
- Mechanized metatheory
- ...