# CSE 505: Programming Languages (Fall '00)

## Instructor: Craig Chambers

- office hours: TuTh 11:30-12:30 (Sieg 309)

## TAs: Todd Millstein and Sorin Lerner

- Todd's office hours: M 2:30-3:30 (Sieg 427)
- Sorin's office hours: TBD (Sieg 433)

## Goals:

- Understand the major concepts in programming languages
- Gain some experience using advanced features of modern programming languages
- Learn to use formal semantics as a tool for understanding and describing language semantics precisely
- Understand trade-offs in language design
- Become aware of current open problems and active research areas in the design of modern programming languages

# Course outline

## Introduction

- history of programming languages
- programming language design principles

## Functional languages

- Scheme
  - atoms, symbols, lists; cons, car, cdr; define, let, let*, cond
  - tail recursion
  - higher-order first-class functions; currying
  - continuations; call/cc
  - syntax extensions
  - meta-circular interpreters; eval
- ML
  - lists, tuples; val, fun, if, let
  - pattern matching
  - type inference
  - polymorphic types
  - higher-order functions
  - datatypes, polymorphic datatypes
  - exceptions
  - references
  - modules; signatures and structures
- Haskell
  - lazy evaluation
  - array & list comprehensions
  - type classes
- Lambda calculus
  - untyped lambda calculus
  - alpha, beta, eta reduction; normal order vs. applicative order evaluation; confluence, normalization
  - simply typed lambda calculus
  - operational semantics; big step/natural semantics; small step semantics
  - soundness, subject reduction
  - polymorphically typed (predicative and impredicative) lambda calculus
  - PCF: extended lambda calculi
    - products, sums, references
  - denotational semantics; domains

# Course outline, cont.

## Object-oriented languages

- Smalltalk
  - objects, classes, instances, methods, instance variables
  - inheritance
  - dynamic dispatching
  - first-class classes; meta-classes
- Pizza
  - subtyping; subtype polymorphism
  - inheritance vs. subtyping
  - covariance vs. contravariance
  - bounded quantification; F-bounded quantification; matching
- Cecil
  - multi-methods
  - multiple inheritance

## Declarative languages (as time permits)

- Prolog
  - clauses
  - unification
  - backtracking
  - cut
- CLP(R)
  - constraints

# Course work

## Readings:

- overviews or reference manuals of languages being studied
- important or fun papers from the literature

## Assignments:

- bi-weekly homeworks
  - thought problems
  - programming projects
- course project

## Grading:

- homework assignments: 25%
- course project: 25%
- midterm: 25%
- final: 25%

# Course project

A major component of the course will be a project that you do, either alone or as part of a pair. The topic of the project is anything that you choose and that I approve. It should, of course, be related to programming languages. Possible projects include:

- the study of a language or major language feature not otherwise covered during the quarter
- a critical analysis of an existing language
- a careful language feature design, extending or modifying an existing language
- the careful design of a new domain-specific language

The project should culminate in a written 10-15 page report, due at the end of the quarter. A guest lecture may also be a possible result of a successful project.