CS 381: Final Exam Review
Overview

Midterm #1
- functional programming
- syntax

Midterm #2
- denotational semantics
- type systems
- naming, scope, parameter passing

Post-midterms
- logic programming
Functional programming

Essential terminology: type, expression, value, function

Kinds of questions:

- what is the type of a given Haskell expression?
- what is the result of evaluating a given expression?
- write a specified value of a given data type
- implement a function given an informal specification

See: Homework #1, Quiz #1, Midterm #1
Syntax

Essential terminology: grammar, abstract syntax, concrete syntax, AST, object language, metalanguage

Kinds of questions:

- what sentences are generated by a given grammar? from which nonterminal?
- encode a grammar as a Haskell data type
- write an object language program as a Haskell value
- draw the AST for an object language program, based on the Haskell value

See: Homework #2, Midterm #1
Denotational semantics

Essential terminology: **semantic domain, valuation function**

Kinds of questions:
- identify the best semantic domain from a syntax + informal spec
- given a syntax + semantic domain, implement the valuation function

See: Homework #3, Homework #4, Quiz #2, Midterm #2
Type systems

Essential terminology: **type, static typing, dynamic typing, typing relation**

Kinds of questions:
- what are some benefits of static typing?
- given a spec + syntax, what is an appropriate representation of types?
- given a syntax + types, implement the typing relation

See: Quiz #2, Midterm #2
Naming and scope

Essential terminology: name, declaration, binding, reference, shadowing, dynamic scope, static scope, environment, closure

Kinds of questions:

- label the declarations and references in a C or Haskell snippet
- what declaration does a name reference?
- evaluate an expression with dynamic vs. static scoping

See: Homework #4, Midterm #2
Parameter passing

Essential terminology: **call-by-value**, **call-by-name**, **call-by-need (lazy)**

Kinds of questions:
- how many times will an argument be evaluated under each scheme?
- what are the properties and tradeoffs of each scheme?

See: (in-class discussion)
Logic programming

Essential terminology: atom, predicate, variable, goal/query, database, fact, rule, rule head, rule body, goal search, unification, cut

Kinds of questions:

- what is the result of a given unification problem?
- given a database, what are all solutions to a given query?
- given a database, write a query that returns a particular result
- given a spec, define a predicate using facts and rules
- given a database, will goal search terminate for a particular query?

See: Homework #5, Quiz #3