CSCI 330: Final Exam Review

This is intended as a guideline for studying for the final. I wouldn't have covered something if I didn't think it was important. If you are wondering about a topic and you don't see it here, ask me!

Sample Types of Questions

- Short answer
- Problem solving: Static vs. dynamic scope; Referencing environments; Analyze effects of (a) operand evaluation order with functional side effects (b) short-circuit evaluation; drawing the stack/activation records for various programs; Shallow binding, deep binding, ad hoc binding with subprogram parameters; Parameter passing modes
- Coding: Recursive types, Folding, all OCaml labs

Chapter 7 (Expressions and Assignment Statements)

- Arithmetic expressions: design issues
  - operators (unary, binary, ternary)
  - operator precedence and associativity
  - operand evaluation order
  - side effects
  - overloaded operators
  - type conversions: narrowing/widening, mixed-mode (coercion), explicit
  - Boolean expressions, short-circuit
- Assignment statements
  - Conditional assignment
  - Compound operators
  - Assignments as expressions
  - List assignments
  - Mixed-mode assignments

Chapter 8 (Statement-Level Control Structures)

- algorithms represented by flowcharts can be coded **only** with two-way selection & pretest logical loops
- Two-way selection statements: design issues
  - Form and type of control expression (arithmetic? Boolean?)
  - Clause form – how is it delimited? Always compound?
  - Nesting selectors
- Multiple-way selection statements: design issues
  - Form and type of control (integer? String? Enumeration?)
  - Is just one selectable segment executed?
  - How are case values specified? Do all values need to be represented?
- Iterative Statements
  - Counter-controlled loops: type and scope of loop variable, can loop variable be changed in body? Are loop variables evaluated once or once every iteration?
  - Logically-controlled loops: pre-test or post-test? Can you transfer out of more than one loop? Can you have multiple entry points?
- Iteration based on data structures
- Unconditional branching
- Guarded commands

Chapter 9 (Subprograms)

- Subprogram fundamentals: definitions, etc.
- Actual/formal parameter correspondence (positional, keyword), default values
- Local referencing environments (stack-dynamic, static local variables)
- Parameter passing modes
- Type checking parameters
- Multi-dimensional arrays as parameters
- Subprogram names as parameters (type-checking, referencing environment)
- Overloaded subprograms, generic subprograms
- Specific design issues for functions

Chapter 10 (Implementing Subprograms)

- General semantics of calls and returns
- Implementing “simple” subprograms – activation records, etc.
- Adding stack-dynamic local variables
    - dynamic link
    - environment pointer
    - call chain
    - local offset
- Nested subprograms:
    - Static scoping – static chain
    - Dynamic scoping: deep access vs shallow access

Chapter 14 (Exception Handling)

- Alternatives to built-in exception handling (how to handle errors in languages w/o exception handling?)
- Advantages to built-in exception handling
- Design issues for exception handling
- Options for continuing after an exception
- What happens with unhandled exceptions

OCaml

- Writing and understanding code with an emphasis on variant types and folding