CSCI 330 (Programming Languages)
Dr. Schwartz
Review Sheet for Exam 3

This is intended as a guideline for studying for the first exam… but only as a guideline! 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, multiple choice and true/false, fill-in-the-blank
- **Problem solving (for example)**
    - **Drawing the stack/activation records for various programs**
    - **Showing how various scoping schemes work -- static and dynamic (deep and shallow), nested subprograms – chap 10**
    - **Shallow binding, deep binding, ad hoc binding with subprogram parameters**
    - **Parameter passing modes**

Chapter 8 (Statement-Level Control Structures)

- all algorithms represented by flowcharts can be coded with only two-way selection and 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 can developers handle errors in languages without exception handling?)
- Advantages to built-in exception handling
- Design issues for exception handling
- Options for continuing after an exception
- What happens with unhandled exceptions

Testing

- Overall process – phases of testing, when test plans should be written
- White box vs black box testing
- Types of testing: Unit testing, integration testing, system testing, regression testing, performance testing, acceptance testing
- Static vs dynamic testing

OCaml
Understanding code with an emphasis on Chapter 14 (Exception Handling)

- Alternatives to built-in exception handling (how can developers handle errors in languages without exception handling?)
- Advantages to built-in exception handling
- Design issues for exception handling
- Options for continuing after an exception
- What happens with unhandled exceptions
- folding