CMSI 585

PROGRAMMING LANGUAGES (GRADUATE LEVEL)

Fall 2006

Midterm Review Sheet

The midterm will take place as scheduled, on October 10. It will be open book, notes, and handouts, but not open computer. This guide should help you to prepare for it properly.

Covered Material

Scott Chapters 1 to 3, and a little of 6.1, are fair game for the midterm, including all handouts and sample code that have been distributed in support of this content (except for the latter part of the *Express Yourself* handout, which, as discussed, have not yet been covered in class).

Sample Tasks and Questions

The following examples represent the types of questions or tasks that you may be asked to accomplish, in addition to the exercises within the textbook:

1. Given code fragments in some programming language, specify what type of error(s), if any, are in the code fragment (i.e., lexical, syntactic, static semantic, dynamic semantic).
2. Describe how simple programs, such as GCD, Roman, and the ones you've written, may vary in different languages, and how they are the same.
3. Be comfortable with both reading and writing regular expressions and context-free grammars: given either one, you should be able to infer what strings may or may not be accepted by either one. Conversely, given a natural language description of a language or token set, you should be able to construct the appropriate specification for that description.
4. Given a syntax design issue (e.g., possible ambiguities, possible parsing or semantic difficulties), provide possible solutions, and be aware of their relative advantages and disadvantages.
5. Given a code fragment in some language, track the lifecycle of the objects and bindings within this code fragment (e.g., when they are created, destroyed, activated, deactivated, etc.) — this includes recognizing whether objects use static, stack, or heap allocation.
6. Given a set of scoping rules (static/lexical vs. dynamic scoping) or rules on when referencing environments are bound (deep vs. shallow binding), track the active bindings within a program (e.g., What is the value of a given variable? To which object is some symbol bound at a given point in the code?).
7. Determine whether a given module implementation or language specification follows the module-as-manager or module-as-type paradigm.
8. Rewrite a given expression using different combinations of prefix, postfix, and/or infix notation. Note how the ability to form expressions in various notations implies some knowledge of the syntax for these notation styles.