# **CS 330 Final Examination Study Guide**

**Time & Date:** 10:00 – 11:40 AM Thursday (5/2) in class room

**Format:**

This test is closed book, notes, and NO electronic device allowed.

You are allowed to bring either three letter-size notecards (8.5x11 each) with hard-written notes on both sides, or one notepaper of any size with hard-written notes on one side.

**Opt-out option:**

You may opt-out of the final exam by taking the letter grade based on your average before final exam with an 8-point deduction. For example, if your average before final is 90, with an 8-point deduction, your average for the course will be 82 and thus resulting in a B grade. If you choose to opt-out of the final exam, you must email me **before** the day of the final exam.

**Material**:

# Scheme (25-30%)

# Prolog (25-30%)

# PLP Chapters 3, 6-8, 10 and 11 (mostly material covered in class, 40-50%)

Be sure to study book chapters, course notes, examples, and HWs.

**Test format**:

1. Matching – basic concepts and terms from textbook.
2. Short answer questions – concepts and methods from the text PLP book, see detailed list below.
3. Scheme

* Trace simple scheme function to determine the return value. This also give you the general syntax of scheme functions.
* Write very simple scheme function using recursion, tail recursion, etc.

1. Prolog

* Given a set of rules and facts, tell the results of a query (Yes/No, variable instantiations, etc.)
* Write rules for simple task/facts to perform simple tasks.

**Specifics:**

PLP Chapters

**Terms and concepts**

Know the following terms:

Binding (static, dynamic), binding time, scope, (static, dynamic), static allocation, frame, activation record, Local Referencing Environment (LRE), heap, fragmentation (external, internal), overloading (function, operator, etc.), side effect, short-circuit evaluation, strong typing, data type (discrete, ordinal, scalar, etc.), type equivalence (name vs. structure), type compatibility, coercion (implicit type conversion), casting (explicit type conversion), row major, column major, lost heap variable dangling pointers, exception handling, co-routine, nameless function, higher order functions, unification, slice or section (of array), co-routine, display, generic subroutine, exception handling, lambda calculus, …

Be able to:

* Describe various binding times and give examples
* Compare and contrast static binding vs dynamic binding and give example for each
* Describe content of a typical frame on the runtime stack given example pascal-like code (similar to the examples in chapter 7)
* Discuss commonly used storage allocation strategies such as best-fit, worst-fit, buddy system, ...
* Describe two most common data structure used by a compiler
* Operator overloading
  + When does it become necessary?
  + Can we overload operator outside of class definition?
* Compare and contrast standard function definition, macro, and inline
* Discuss static scope implementation methods (static chain vs. display)
* Describe the content of runtime stack during a point of execution, including both static and dynamic links.
* Discuss the benefit of short circuit evaluation with an example
* Describe the basic components of a type system
* Discuss the difference between ordinal type and scalar type
* Describe the two major approaches in determining type equivalence
* Discuss the two layout strategies for array implementation (contiguous elements vs. row pointers)
* Discuss two common problems with pointer variables
* Discuss methods for heap management
* Describe the concept of generic subroutine and illustrate with an example
* Compare and contrast static vs dynamic scope. Discuss methods for implementation of static scope (static chain vs display), and dynamic scope (deep access vs shallow access)
* Know who created some of the important systems, models, and languages in CS such as lambda calculus, UNIX, Turing Machine, Fortran, …

Scheme

* Important functions and operators:

and, append, begin, car, cdr, c????r, cond, cons, define, display, do, equal?, eval, if, lambda, let, list, list?, equal?, member, not, null?, or, +, -, \*, /, =, …

* Know the basic structure of lists, sub-list, and be able to use the appropriate procedures to manipulation list.
* Be able to analyze simple scheme functions and determine what a function call returns.
* Understand recursion and be able to write recursive function to do simple things similar to those of homework.
* Understand tail recursion technique and its benefit.

Prolog

* Know basic prolog functors and data structures
* Be able to trace and tell the results of simple prolog queries, and identify multiple solutions for a query.
* Be able to write prolog rules that perform simple task including list processing.
* Discuss/describe how a prolog implementation system such as swi-prolog go about searching answer for a goal (hint: unification). Illustrate with an example.