FINAL EXAM STUDY GUIDE

This exam is *comprehensive* and will cover all of the material presented in class. The exam will place an emphasis on the final part of the semester; including logic programming with Prolog. Sample questions are listed below. The sample questions do not necessarily cover all of the content of the exam but they are representative of the types of questions that will be asked.

- List the three types of memory utilized by imperative programs and the differences between them.
- Define the terms: strongly/weakly typed, type inference, high-order function, dynamic/static typing, lazy/eager evaluation, unification, heap memory, stack memory, static memory, call by value/value-result/reference.
- Describe how a C macro works and why macros are supported.
- What features characterize a(n) object-oriented/imperative/logical/functional language?
- Does Scheme/Haskell/Java use lazy or eager evaluation?
- What is an enumeration? How are enumerations in Java superior to enumerations in C?
- Write a "length", "reverse", "member", or "filter" method in Scheme/Haskell. If using Scheme, use only the four fundamental list processing functions and conditionals.
- What is the difference between pre-defined words and reserved words?
- Define the difference between row-major and column-major layout for multi-dimensional arrays. How does layout differ from allocation? What layout scheme does Java use?
- How are patterns used in Haskell and does this usage differ from the way that functions are overloaded in Java?
- Write a "last" function/predicate in Haskell/Prolog that accepts a list of elements and returns the last element of the list.
- Write a list-append function in Prolog/Scheme/Haskell.
- Write a count function in Prolog that accepts a list and an element and counts the number of occurrences of the element in the list.
- Show that the following grammar is ambiguous: <e> := <e>+<e> | <e>*<e> | **A** | **B**
- Write a "member" predicate in Prolog which determines whether an element E is a member of list X.
- Write a "first" predicate in Prolog that determines the first element in a list.
- Consider the Haskell statement: data Btree a = Empty | Node a (Btree a) (Btree a). Write functions that 1) find the height of a binary tree, 2) find the size of a binary tree, 3) determine if a binary tree is a binary search tree, 4) give the pre-order listing of the elements of a binary tree.