CS 335 - Exam 2 Study Suggestions

last modified: 4-5-11

- You are responsible for all material covered in lectures and homeworks; this is just a quick overview of especially-important material.
- You are permitted to bring into the exam a single piece of paper (8.5" by 11") on which you have handwritten whatever you wish on one or both sides. This paper must include your name, it must be handwritten by you, and it will not be returned.
  - Other than this piece of paper, the exam is closed-note, closed-book, and closed-computer.

Lisp/Scheme/Functional Programming

Lisp/Scheme

- You should know the basics (including the syntax and semantics) of `let`, `let*`, and `letrec`. Why would you use these? What is the difference between these three forms? You should be able to give the value of a given `let`/`let*`/`letrec` expression.

names, bindings, and scope

- What are some of the attributes that can be associated with a name?
  - Given a statement, be able to give/describe attributes that are associated with a name as a result of that statement;
- What is meant by binding? What is meant by the binding time of an attribute?
  - What is static binding and dynamic binding?
  - What is a static attribute? What is a dynamic attribute?
  - Remember: we usually refer to the binding time of the attribute as the earliest time that the language rules permit the attribute to be bound.
  - You should be able to reason about/determine whether an attribute is a static attribute or a dynamic attribute.
- What were the 4 stages of execution for early FORTRAN? What happens in each? Within that first stage, what were the 3 phases? What happens in each?
- What are the subcategories of static binding?
- You should be able to reason about/determine which subcategory a static binding belongs in, or that a binding is dynamic; you should be able to give examples of bindings in each of the static binding subcategories and in the dynamic category
- In general, what are the tradeoffs between earlier and later binding times?
- What is a symbol table? What can it be thought of as doing?

- What is meant by the scope of a binding? What is static scoping? dynamic scoping?
  - you should be able to tell the values of variables in a fragment of code based on whether static or dynamic scoping is being used;

Prolog and logic programming

- Note that logic programming languages are neither imperative nor functional;
  - and, in logic programming the What (logic) is separated from the How (control)
  - the computation engine is based on theorem-proving and recursion;
- from what area of research/work did Prolog arise?
- theorem-proving based on what is the foundation of Prolog? (resolution)
- what is a Prolog fact? rule/predicate? query? should be able to write/read all three of these, and be able to reason about them as Prolog would;
- what is a Horn Clause? what does it mean? what does a Horn Clause look like in Prolog?
  - what part of a Horn Clause is its consequent, goal, or head?
  - what part of a Horn Clause is its antecedents, subgoals, or tail?
  - what is a Horn Clause (in Prolog) with no tail also called?
  - what is a Horn Clause (in Prolog) with a tail also called?
- how does Prolog (including SWI-Prolog) tell that something is a variable?
- what is an auxiliary variable in Prolog?
- what is the scope of a variable in Prolog?
- what goes in a Prolog knowledge base? How do you load that knowledge base in SWI-Prolog?
- how can you create a record (in a separate file) of what has gone in in a SWI-Prolog session?
- what are some ways in which Prolog differs from "pure" logic programming?
- how can you express a list in Prolog? You should be comfortable using lists within Prolog.
- what is unification? resolution? backward-chaining? backtracking?
  - how does one write a recursive predicate/rule in Prolog?
  - in SWI-Prolog, how can you "follow" the resolution process step-by-step?
- how do you *typically* "repeat" in Prolog?
- what is a cut? what are its semantics and side-effects? why is it useful?
- how can cut and fail be used to define the concept of negation in Prolog (negation by failure)?
- what are the 3 main common uses of cut, according to Clocksin and Mellish?
  - to tell the Prolog system that it has picked the "correct" rule for a particular goal;

  - to tell the Prolog system to fail a particular goal immediately, without trying for alternative solutions;
  - when we want to terminate the generation of alternative solutions through backtracking;
- should be comfortable with recursion in Prolog; regardless of the language, what are the essential features of "proper" recursion?
- do need to be comfortable with the essentials of the Prolog/SWI-Prolog language;
  - how are Boolean operators (and, or, not) written?
  - what does = mean in Prolog? What does == mean?
  - how does one handle lists in Prolog? What are Prolog's alternatives to Scheme's car and cdr?
  - what does the `protocol` predicate cause to happen, in a SWI-Prolog session?
  - how is the `is` operator used with arithmetic expressions? what is its effect?
  - should be comfortable with SWI-Prolog's `assert`, `retract`, `write`, `writeln`, and `nl` predicates;
- you will have to read/write Prolog code; given a Prolog knowledge base, you may have to give the results for queries on that knowledge base.

object lifetimes, storage allocation mechanisms, automatic vs. manual storage reclamation, basics of garbage collection

- What is meant by a binding's lifetime? What is meant by an object's lifetime? Do these need to necessarily coincide? What is an example where it is reasonable if they do not? What is an example where it is an error if they do not?
- What are the three principal storage allocation mechanisms? How are objects allocated and deallocated in each? You should be able, also, to give examples of objects that would be allocated using each of these storage mechanisms.
- What is the difference between the heap data structure and the heap storage mechanism? What is allocation using a heap storage mechanism usually called?
- What is meant by manual storage reclamation? by automatic storage reclamation? Why is storage reclamation needed?
  - What are some advantages of manual storage reclamation? some disadvantages?
  - What are some advantages of automatic storage reclamation? some disadvantages?
- What is garbage collection? What is meant by "garbage-collected languages"?
  - note that some languages require garbage collection, either as part of the language specification or just for practical considerations; some were designed for use with manual memory management, although even these might have garbage collected implementations; and some allow for both manual and automatic storage reclamation, with variations;
  - you should know some examples of languages in each of these categories;