CS 335 - Exam 1 Study Suggestions

last modified: 2-16-11

- 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.

Formal Languages

Regular Expressions (RE's)

- What is a Regular Expression (RE)? What can they be used for? For what purposes are they often used in programming languages?
- You should be able to read, write, and answer questions about RE's.
- What is meant by the alphabet of a RE?
- You should be able to read/understand the notation { a | b } for describing a language.
- Given an RE R, you should be able to describe L(R) (the language described by that RE). You should be able to give example strings in L(R); you should be able to determine whether a given string is in L(R).
- Given a description of a regular language and its alphabet, you should be able to give an RE describing that language.

Context-Free Grammars (CFG's)

- What is a Context-Free Grammar (CFG)? What can they be used for? For what purposes are they often used in programming languages?
- You should be able to read, write, and answers questions about CFG's.
- What is a variable/nonterminal? What is a terminal? What is a start symbol? What is a production?Given a CFG G, you should be able to determine what the nonterminals, terminals, start symbol, and productions are for that CFG.
- L(G) means the language generated by a CFG G. How can you prove that a string is in L(G)?
- What is a derivation? What does it mean? Given a CFG, you should be able to write a derivation in the format discussed in class. What is the significance of there being a derivation for a particular string?
- What is a leftmost derivation? What is a rightmost derivation? You should be able to write such derivations given a CFG.

Backus-Naur Form (BNF)

- What is BNF?

  - Who is it named after? During the development of what programming language was BNF developed? Why was it developed? What is BNF used for? What are the advantages of using BNF for that purpose?
- What is the relationship between CFG's and BNF's?
- You should be able to identify the terminals, nonterminals, and productions of a given BNF.
- You should be able to read, write derivations based on a given BNF.
  - What is the significance of a string having or not having a derivation based on a given BNF?
  - Based on a given BNF, you should be able to explain why a derivation might not be possible for a given string.
- You should be able to read and understand a BNF; you may have to write a BNF.
- What is EBNF? What is added to EBNF? Can EBNF describe more languages than BNF?

Parse trees

- What is a parse tree/derivation tree? You should be able to read, create, and answer questions about these. What does a parse tree show?
- Given a CFG G, or a BNF, or an EBNF, you should be able to draw a parse tree for a string in that grammar's language; you might have to draw a parse tree for a particular string in that grammar's language.
- When is a grammar (whether a CFG, a BNF, or an EBNF) said to be ambiguous? How can you prove that a grammar is ambiguous? Why might it be undesirable for a grammar for a programming language to be ambiguous?
  - You may be asked to prove that a given grammar is ambiguous.

Lisp/Scheme/Functional Programming

Lisp/Scheme

- What led John McCarthy to start developing Lisp? What is the name Lisp an acronym for?
- What control structure did McCarthy need for his work, develop, and suggest be included in Algol (as well as including it in Lisp)? He wanted to use this with what powerful concept for processing lists?
- Why was it important to McCarthy that the conditional expression be short-circuiting? What is meant by "short-circuiting", in this context?
- What is meant by the term "garbage collection"? Why did McCarthy want this for Lisp?
- How did Lisp's distinctive notation come about? Why did it persist?
- How are both data and programs represented in Lisp?
- What is the relationship between linked lists and Lisp lists?
- You should expect to have to translate algebraic expressions into Scheme expressions; you should be able to read and write such expressions.
- You should be quite comfortable with Lisp's basic list operations of `cons`, `car`, and `cdr`. How are they

implemented using linked lists? You should be comfortable with `append`, `list`, `null?`, and `list?`, as well.

  - How can an expression consisting of several composed `car`/`cdr` calls be abbreviated as a single function call?

- How is repetition typically accomplished in Lisp and Scheme?
- Which is more powerful, in terms of what one can compute: recursion or iteration?
- What is a higher-order function?
- What is an anonymous function?
- You should be comfortable with lambda expressions; you should be able to read and write them, including being able to explain what a given lambda expression represents.
- You should expect to have to read and write Scheme list expressions; you should expect to have to read and write Scheme functions, including higher-order functions. (Recursion will be involved!) You should be comfortable with the conditional expression, basic predicates, and the features used in Homeworks 2 and 3.
- You should be able to read and write fragments of Scheme code, including defining Scheme functions.

Pure Functional Programming

- What are some of the ways that pure functional programming differs from procedural/imperative programming?
- What operation is not available in pure functional programming? What common control structure is thus also not available in pure functional programming? What takes the place of this control structure in pure functional programming?
- Is pure functional programming Turing complete?
- What is meant by the term referential transparency?
- In pure functional programming, we say that functions are first-class objects, and we can write so-called higher-order functions. What is meant by saying that a function can be a first-class object? What is a higher-order function?
- In lecture (and posted) we discussed five qualities of functional programming languages and functional programs (as summarized by Louden) - what are they?
- What are some of the benefits of functional programming (and of the functional style of programming, even within imperative languages)?
- Are Lisp and Scheme suitable only for pure functional programming? Are they used only for pure functional programming?
- What is meant by the term **currying**?