Concepts of Programming Languages, CSCI 305, Fall 2021
Review for exam 1, Sept. 24

Introduction, Chapter 1

- Know various ways that languages are categorized
- Know the difference between procedural/imperative languages and declarative/non-imperative languages and some examples of each.
- Be able to contrast the process of assembling, compiling, and interpreting, and that modern languages do a combination
- Know how the following fit into language translation
    - scanner, parser, semantic analysis and intermediate code generation
    - character stream, token stream, parse tree, abstract syntax tree
- Know the difference between the front and back end of language translation and how one front end can be used with different back ends, and vice-versa.

Programming Language Syntax, Chapter 2, pages 43-69

Section 2.1 Specifying Syntax: Regular Expressions and Context-Free Grammars
Section 2.2 Scanning

- Be able to explain where errors will be captured in the compilation process: lexical analysis (scanning), syntactic analysis (parsing), semantic analysis, dynamic semantic analysis or compiler can't catch.
- Know the difference between syntax and semantics, and be able to give examples of each in the context of program translation.
- Know the tasks of a lexical analyzer.
- Know the difference between a lexeme and a token.
- Know the definition of regular expressions
- Be able to convert:

Language description ⇒ Reg-ex
Reg-ex ⇒ Language description

- Given pattern descriptions for tokens, be able to give regular expressions and productions for recognizing them.
- Know the definition of NFAs and DFA
- Be able to do the following conversions using the method given in the text and discussed in class:

Reg-ex ⇒ NFA
NFA ⇒ DFA
DFA ⇒ Minimal DFA

Functional Languages, Chapter 11, 535-550, 581-584

Section 11.1 Historical Origins
Section 11.2 Functional Programming Concepts
Section 11.3 A Bit of Scheme
Section 11.8 Functional Programming in Perspective
Section 11.9 Summary and Concluding Remarks

- Know that Scheme came from LISP which stands for LISt Processing (defined by John McCarthy in 1958)
- Know the meaning of referential transparency and that pure functional languages have referential transparency
- Know that Scheme operates on lists and know how these lists are stored
- Know the following Scheme built-in functions:
  car, cdr, cons, list, append, length, define, lambda, if, cond, member, assoc, let, begin
- Be able to write simple programs in Scheme