Date and time of Final _____

***This is part 2 of the study guide for the final. Part 1 is the study guide for the midterm.***

**Basic Topics**

- *Chapter 1*: Introduction
  - Organization of a compiler
    - Lexical Analyzer (Scanner)
    - Syntactical Analyzer (Parser)
    - Semantical Analyzer (Checker)
    - Optimizer
    - Code Generator
      - Cross Compiler
      - Native Code Compiler
  - Automated Tools
    - Lex
    - Yacc
  - Basic data structures of a compiler
    - Tokens
    - Symbols Table
    - Abstract Syntax Tree
- *Chapter 2*: Scanning
  - Regular Expression
  - Extensions to Regular Expressions
  - Deterministic Finite Automaton
  - Non-Deterministic Finite Automaton
  - Thompson's Construction
  - Subset Construction
- *Chapter 3*: Context-Free Grammars and Parsing
  - Terminals and Non-Terminals
  - Productions
  - Derivations: Left-Most and Right-Most
  - BNF and EBNF
  - Parse Tree and Ambiguous Grammars
  - Precedence and Associativity
  - Disambiguating Rule
- *Chapter 4*: Top-Down Parsing
  - Removing Left Recursion
  - Left Factoring
  - Predictive Parsing
    - Recursive Descent
    - LL(1)
  - First and Follow sets

- *Chapter 5*: Bottom-Up Parsing
  - LR(0) Items
  - LR(0) DFA
  - LR(0) Parsing
  - SLR(0) Parsing
  - LR(1) Parsing
  - LALR(1) Parsing
- *Chapter 6*: Semantic Analysis
  - Atrributes
    - Inherited
    - Synthesized
  - Attribute Grammars
  - The Symbols Table

1. Describe what is meant by an LR(0) item. What is meant by an *initial item*? What is meant by a *completed item*?

2. Given the grammar

   $$L' \rightarrow L$$
   $$L \rightarrow ( L ) \mid a$$

   List the LR(0) items for this grammar

3. Given the grammar in question 2, draw the LR(0) NFA for the grammar.

4. Given the NFA in question 3, draw the LR(0) DFA.

5. Given an LR(0) DFA, what is a *kernel* item? What is a *closure* item? Give an example of each type of item using the DFA in question 4.

6. Given the LR(0) DFA in question 4, draw the paring table for the grammar in question 2.

7. Given the LR(0) parsing table in question 6, show the parsing action for the string ( ( a ) ).

8. Bottom-up parsers are often described as *Shift-Reduce* parsers. What does this mean?

9. In describing a bottom-up parser, what is meant by a *shift-reduce* error? What is meant by a *reduce-reduce* error?

10. For what does the acronym SLR stand? Describe how SLR(1) parsing differs from LR(0) parsing.

11. Which is the more powerful parsing algorithm: SLR(1) or LR(1)? Explain your answer.

12. How does an LR(1) item differ from an LR(0) item? What do the numbers in the parentheses represent?

13. For what does the acronym LALR stand? Describe how an LALR(1) parser differs from an LR(1) parser.

14. Of the bottom up parsing algorithms—LR(0), LR(1), SLR(1), and LALR(1)—which is the most powerful? Which is the algorithm implemented by automated tools such as YACC?

15. Given a parsing table such as the one in table 5.10 on page 222 of the textbook, show the action of parsing a given string.

16. Given a DFA such as the one in figure 5.8 on page 223 of the textbook, create the parsing table.

17. Given a programming language construct, what is meant by an *attribute*?

18. Describe the difference between *syntax* and *semantics*.

19. What is meant by phrase *syntax driven semantics*?

20. What is meant by the *binding time* of an attribute? What is the difference between *static* binding and *dynamic* binding of an attribute? What might be an example of each type of binding?

21. What is meant by an *inherited* attribute? What might be an example of an inherited attribute? How are inherited attributes calculated?

22. What is mean by a *synthesized* attribute? What might be an example of a synthesized attribute? How are synthesized attributes calculated?

23. What is a *symbols table*? Why might a semantic analyzer use the symbols table? What are some of the ways to *implement* a symbols table?

24. In the context of a programming language, what is a *block*? When a language is block structured, the implementation of a symbols table can be complex. Why is this?