Name: ______________________________

Exam #1: Part 1 of 2

COMP.3010 – Organization of Programming Languages; March 2, 2018 – Dr. Wilkes

Note: *This exam is closed book and notes, except for one 8.5x11" sheet of paper with handwritten notes (no photocopies).*

Multiple Choice Questions – 5 points each: Circle the correct answer.

1. Which stage of a compiler reads a stream of characters and produces a stream of tokens?
    a. Code generator
    b. Code improvement stage
    c. Parser
    [✓ d.] Scanner
    e. Semantic analysis stage
2. Which stage of a compiler determines the meaning of a program, and checks for the violation of rules such as type errors and "declaration before use"?
    a. Code generator
    b. Code improvement stage
    c. Parser
    d. Scanner
    [✓ e.] Semantic analysis stage

    Symbol table: collects declaration and scope information to satisfy " declaration before use " rule, and to establish data type and other properties of names in a program.
3. **(CIRCLE ALL THAT APPLY)** If a grammar is ambiguous, what type of grammar can it be?
    a. LL
    b. LR
    c. LALR
    d. SLR
    [✓ e.] None of the above
4. **(CIRCLE ALL THAT APPLY)** If a grammar can be implemented using a recursive descent parser, what type of grammar can it be?
    [✓ a.] LL
    b. LR
    c. LALR
    d. SLR
    e. None of the above
5. **(CIRCLE ALL THAT APPLY)** If a grammar contains left-recursive productions, what type of grammar can it be?
    a. LL
    [✓ b.] LR
    [✓ c.] LALR
    [✓ d.] SLR
    e. None of the above

    LL grammars cannot exhibit left-recursive productions (but LR can)
    Example

Name: ______________________________
https://www.chegg.com/homework-help/questions-and-answers/18-ada-solves-dangling-else-problem--using-simple-syntax-ambiguous-grammar-external-rule-b-q27135383

6. In Modula-2 and Ada, the "dangling `else`" problem was addressed by:
   [✓ a.] Adding an "end marker" for the `if` statement to the grammar
   b. Including a semantic rule in the language reference manual stating that an `else` clause matches the closest unmatched `if` clause
   c. Using indentation to indicate which `else` clause matches which `if` clause
   d. None of the above
7. **(CIRCLE ALL THAT APPLY)** Phases that are commonly associated with the "front end" of a compiler include:
   a. Machine-independent code generator
   b. Machine-independent code improvement stage
   c. Machine-dependent code generator
   d. Machine-dependent code improvement stage
   [✓ e.] Parser
   f. Scanner
   g. Semantic analysis stage
8. ~~Which of the following is an example of a "high-level" intermediate form?~~
   [✓ a.] ~~Abstract syntax tree~~
   b. ~~Register Transfer Language (RTL)~~
   c. ~~Stack machine instructions (e.g., Pascal P-code or Java bytecode)~~
   d. ~~Three-address instructions (quadruples)~~

True/False Questions – 2 points each: Write T or F beside each question.

F 1. On modern machines, assembly language programmers still tend to write better code than compilers can generate.
For modern microprocessor architectures, particularly those with so-called superscalar implementations (ones in which separate functional units can execute instructions simultaneously), compilers can usually generate better code than can human assembly language programmers.

T 2. Backus-Naur Form (BNF) originally was devised to enable the formal specification of the grammar for the Algol-60 programming language.

F 3. Operator precedence can be specified in a grammar by incorporating productions which cause the higher-precedence operators to appear "higher" (closer to the root node) in the parse tree of an expression.

T 4. A scanner must sometimes "peek" at upcoming characters.

F 5. Epsilon productions are more common in LR grammars than in LL grammars.

T 6. Assuming that there are no epsilon productions in the grammar, a top-down parser will predict the production $A \rightarrow \alpha$ if the current token $T \in FIRST(\alpha)$.

F 7. An *S-attributed* grammar may contain both synthesized and inherited attributes.

T 8. A recursive descent parser may incorporate semantic attributes via action routines.

F 9. ~~Most front ends for the GCC compiler use an RTL-based intermediate form.~~

F 10. ~~A machine-dependent code generation phase can assume that an unlimited number of registers are available.~~

Name: ________________________________

Exam #1: Part 2 of 2

COMP.3010 – Organization of Programming Languages; March 2, 2018 – Dr. Wilkes

Note: *This exam is closed book and notes, except for one 8.5x11" sheet of paper with handwritten notes (no photocopies).*

Short Answer Questions – 10 points each: Write your answer in the space provided.

1. Errors in a computer program can be classified according to when they are detected and, if they are detected at compile time, what part of the compiler detects them. Using your favorite C-family language (C, C++, or Java), give an example of each of the following:
   a. A **lexical error**, detected by the scanner
   b. A **syntax error**, detected by the parser
   c. A **static semantic error**, detected by semantic analysis
   d. A **dynamic semantic error**, detected by code generated by the compiler
   e. An error that the compiler can neither catch, nor easily generate code to catch

```
a. String 1sentence = "hello world";
b. String myString = "hello
                                world";
c. int notADouble = 2.5;
d. int[] myArray = new int[10];
   array[10] = 100; // out of range
e. for ( int ctr = 0; ctr > 100; ctr++)
   { ctr = set_zero(); }
   int set_zero() {return 0;}
```

2. Describe the difference between an *S-attributed* grammar and an *L-attributed* grammar, and the circumstances in which each may be used.

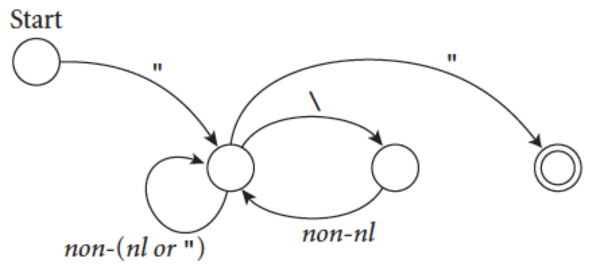
Simply S-attributed Grammar is the Grammar who has strictly Synthesized type of grammar means Only having Value attribute throughout the parse tree

where as L-Attributed grammar can have both synthesized as well as Inherited grammar with some of the rules like one having the transfer of inheritance from always left to right.

S-attributed SDT :
If an SDT uses only synthesized attributes, it is called as S-attributed SDT.
S-attributed SDTs are evaluated in bottom-up parsing, as the values of the parent nodes depend upon the values of the child nodes.
Semantic actions are placed in rightmost place of RHS.
L-attributed SDT:
If an SDT uses both synthesized attributes and inherited attributes with a restriction that inherited attribute can inherit values from left siblings only, it is called as L-attributed SDT.
Attributes in L-attributed SDTs are evaluated by depth-first and left-to-right parsing manner.
Semantic actions are placed anywhere in RHS.

Name: ______________________________

3. C-strings can be defined using the DFA "circles and arrows" diagram shown below. Draw a similar DFA diagram for comments in Pascal. These are delimited by (* and *) or by { and }. They are not permitted to nest.



4. Consider a language like Ada or Modula-2, in which a module *M* can be divided into a specification (header) file and an implementation (body) file for the purpose of separate compilation. Should *M*'s specification itself be separately compiled, or should the compiler simply read it in the process of compiling *M*'s body and the bodies of other modules that use abstractions defined in *M*? Explain the tradeoffs of your choice. If the specification is compiled, what should the output consist of?