Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**COMPILERS**

**Final Exam**

July 23, 2009

1. **(1 Point) The grammar S🡪 aSbS |  is**
2. Left recursive
3. LL(1)
4. Ambiguous
5. None of these
6. **(1 Point)**  **The grammar S🡪 aSb | a is**
7. Left recursive
8. LL(1)
9. Ambiguous
10. None of these
11. **(1 Point) The grammar S🡪 Sab |**  ** is**
12. Left recursive
13. LL(1)
14. Ambiguous
15. None of these
16. **(1 Point)**  **For the grammar S🡪 a S T | , T 🡪 a T b | , the handle of  *a S a a T b b* is**
17. a S a
18. a T b
19. a S a a T b
20. None of these
21. **(1 Point) If the items A 🡪 a • c and S 🡪 c • occur in the same state, there is**
22. A shift-reduce conflict
23. A shift-reduce conflict only if c  FOLLOW (A)
24. A shift-reduce conflict only if c  FOLLOW (S)
25. A reduce-reduce error only if c  FOLLOW (A)
26. **(1 Point) Synthesized attributes are computed by**
27. The scanner
28. Ascending the tree
29. Descending the tree
30. None of these
31. **(1 Point) Loops are**
32. Cycles
33. Cycles whose tails dominate their heads
34. Cycles whose heads dominate their tails
35. Cycles with more than one entry
36. None of these
37. **(1 Point) Static links show**
38. Scoping
39. The previous activation record
40. The return point in the code
41. None of these
42. **(1 Point) For a given grammar G with alphabet , L(G) =**

1. {w **** \* | S 🡺 w}
2. {w **** \* | w\* = w}
3. {w **** \* |  is in w}
4. {w **** \* | length(w) = n}
5. **(1 Point) For S 🡪 a S a | b S b | , L(G) =**
6. (a U b )\*
7. (0 U 1 )\*
8. {w | number of a’s in w = number of b’s in w}
9. Strings of a’s and b’s that read the same forward as backward
10. None of these

**11. (3 Points) Show that the following grammar is *ambiguous* using the given sentence as an example.**

Sentence: **a + b \* c**

Grammar:

T  P | *id* | T ‘\*’ T

P  T ‘+’ T

**12. Consider the following grammar**:

S 🡪 A a | b A c | d c | b d a

A 🡪 d

a) **(5 Points)** ***Using the definition*** of LL(1), show whether this grammar is LL(1) or not

b) **(5 Points)** Show whether this grammar is SLR(1)

**13. Consider the following context-free grammar that generates regular expressions over the alphabet {a, b}.**

a) **(5 Points) Add attribute(s) and semantic functions that compute the maximum number of nested kleene star operators. For example, (a)\* | ((b)\* | a)\* has depth 2. The semantic actions for the 3 base cases are given.**

**Grammar Semantic Actions**

R 🡪 **a**  R.*depth* = 0

 |  **b** R.*depth* = 0

 | **** R.*depth* = 0

 | R1  **|** R2

 | (R1 )\* | (R1 )

 | (R1 ) \*

 | (R1 )

b) (2 Points) Show a parse tree (don’t worry about being fancy) and evaluate attributes for (a)\* | ((b)\* | a)\*

14. (9 Points) Using the following grammar

S 🡪 a S | b

**Create an interpreter using lex and yacc which will count the number of *a’*s in the input string*.* Do not worry about *minor* syntax errors:**

1. **Lex file**
2. **Yacc file with semantic actions (the $$ stuff) to compute the number of *a*’s.**

**15. (3 Points)**

