CSC 348: Introduction to Compiler Design: 2017 Fall

Assignment #2

Last Modified 2017 September 20

Purpose:

To go over:

- Regular Expressions, Non-deterministic Automata, and Deterministic Automata
- The Unix tool Lex

Assignment:

1. **Regular Expressions and Deterministic Automata:**
    1. (10 Points) Write a regular expression to recognize strings have any number of characters in $(a|b|c)^*$, but *must* end with cc.

■ **Positive examples:**

■ **String**	■ **Comment**
■ cc	■ The simplest string in the language
■ abcc	■ ends with cc
■ ccc	■ ends with cc
■ bcc	■ ends with cc

■ **Negative examples:**

■ **String**	■ **Comment**
■ (The empty string)	■ No cc
■ ab**c**	■ Only one ending c

■ ccb	■ cc is not at end

2. (10 Points) Write a *Deterministic Finite Automaton* that can recognize the language above.

2. **Regular Expressions, Non-deterministic Automata, and Deterministic Automata:**
3. Consider the regular expression: $(a|b)^*(ab|ba)$
    1. (20 Points) Convert this Regular Expression into a *Non-deterministic Automata*. **Show work!**
    2. (20 Points) Convert your *Non-deterministic Automata* into a *Deterministic Automata*. **Show work!**
4. **Lex and Flex:**
5. (40 Points)
6. Recognize the tokens of the Pascal programming language.
7. When you see the pattern on the left, output its name as given on the right. Each token gets its own line. Integers, string constants and identifiers should also have their values printed (*What was the name of the C char array that contains the text?*) Do **not** worry about floating points.

8. **Pattern**	9. **What to print**
10. space, tab, newline, carriage return	11. Ignore (no output)
12. { . . . }	13. Ignore Pascal-style comments
14. (* . . . *)	15. Ignore Pascal-style comments
16. [a-zA-Z_][a-zA-Z_0-9]*	17. IDENTIFIER (print its value)
18. [0-9]+	19. Integer (print its value)
20. NIL or Nil or nil	21. NIL
22. ' . . . '	23. STRING CONSTANT (print its value)
24. :=	25. ASSIGNMENT
26. =	27. EQUALS
28. ;	29. SEMICOLON
30. ,	31. COMMA

32. :	33. COLON
34. ..	35. RANGE
36. .	37. PERIOD
38. (	39. BEGIN PAREN
40.)	41. END PAREN
42. [	43. BEGIN BRACKET
44.]	45. BEGIN BRACKET
46. PROGRAM or Program or program	47. PROGRAM
48. CONST or Const or const	49. CONST
50. TYPE or Type or type	51. TYPE
52. VAR or Var or var	53. VAR
54. PROCEDURE or Procedure or procedure	55. PROCEDURE
56. FUNCTION or Function or function	57. FUNCTION
58. BEGIN or Begin or begin	59. BEGIN
60. END or End or end	61. END
62. IF or If or if	63. IF
64. THEN or Then or then	65. THEN
66. ELSE or Else or else	67. ELSE
68. NOT or Not or not	69. NOT
70. AND or And or and	71. AND
72. *OR* or *Or* or *or*	73. OR

74.	REPEAT or Repeat or repeat	75.	REPEAT
76.	UNTIL or Until or until	77.	UNTIL
78.	FOR or For or for	79.	FOR
80.	TO or To or to	81.	TO
82.	DOWNTO or Downto or downto	83.	DOWNTO
84.	WHILE or While or while	85.	WHILE
86.	DO or Do or do	87.	DO
88.	WITH or With or with	89.	WITH
90.	CASE or Case or case	91.	CASE
92.	OF or Of or of	93.	OF
94.	<	95.	LESS THAN
96.	<=	97.	LESS THAN EQUAL TO
98.	>	99.	GREATER THAN
100.	>=	101.	GREATER THAN EQUAL TO
102.	<>	103.	NOT EQUAL TO
104.	+	105.	PLUS
106.	-	107.	MINUS
108.	*	109.	MULTIPLICATION
110.	/	111.	DIVISION
112.	DIV or Div or div	113.	INTEGER DIVISION
114.	MOD or Mod or mod	115.	MOD
116.	IN or In or in	117.	IN
118.	PACKED or Packed or packed	119.	PACKED

120. ARRAY or Array or array	121. ARRAY
122. SET or Set or set	123. SET
124. RECORD or Record or record	125. RECORD

126. Please read the file cs3.pas and generate an output file that is the names of the lexemes that are present. The list of lexeme names that you should generate is in cs3.lexemes. Use the Unix tool diff to compare your output with the "correct" output in cs3.joe.lexemes:
127. $ ./pascalLex < cs3.pas > cs3.lexemes
$ diff cs3.lexemes cs3.joe.lexemes

128. If diff has no output then the two files are equivalent.