ECE 468 — Midterm 1

September 30, 2010

Name: _____

Purdue email: _____

Please sign the following:
I affirm that the answers given on this test are mine and mine alone. I did not receive help from any person or material (other than those explicitly allowed).

X _____

Part	Points	Score
1	8	
2	20	
3	10	
4	22	
5	40	
Total	100	

Part 1: Short answers (8 points)

1) Explain (in *at most* 30 words) why it is useful to split the phases of a compiler into a *front end* and a *back end* (2 points)

Splitting the front and back ends allow us to reuse the front end with different backends (for different ISAs) or reuse the back end with different front ends (for different languages)

2) Give an example of an English sentence that is *syntactically* correct but not *semantically* correct. (1 point)

Colorless green ideas sleep furiously, or many other options.

3) Briefly explain (it shouldn't take you more than two sentences) why the following grammar is not LL(k) for any k (3 points)

S → E $
E → int
E → (E + E)
E → (E – E)

To decide between (3) and (4), we need to know if a '+' or a '–' is coming. But we can see any number of '('s before the operator, so no matter how much look ahead we have, we cannot distinguish between the two productions.

4) Give a grammar that is equivalent to the grammar in problem 3, but *is* LL(1) (2 points)

S → E $
E → int
E → (E O E)
O → + | –

or other, equivalent grammar.

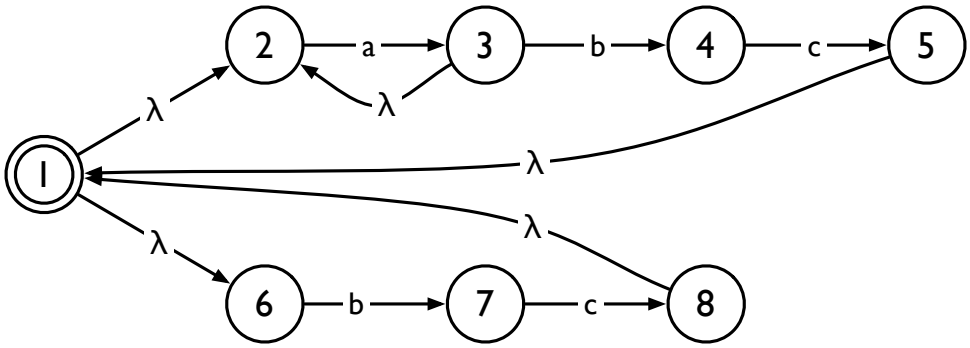
Part 2: Regular expressions, finite automata and scanners (20 points)

1) Describe, in one sentence, the strings captured by the following regular expression (2 points):

$$(ac|ab)^*$$

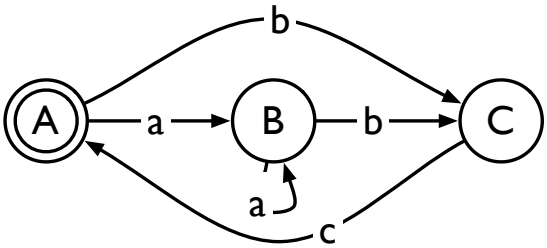
'a' followed by 'b' or 'c', repeated 0 or more times

2) Consider the following non-deterministic finite automaton. Fill in the transition table of its deterministic equivalent. (14 points)



State	Final?	a	b	c	New Name
{1, 2, 6}	y	{2, 3}	{7}	error	A
{2, 3}	n	{2, 3}	{4}	error	B
{7}	n	error	error	{1, 2, 6, 8}	C
{4}	n	error	error	{1, 2, 5, 6}	D
{1, 2, 6, 8}	y	{2, 3}	{7}	error	E
{1, 2, 5, 6}	y	{2, 3}	{7}	error	F

2) Draw the *reduced* version of the DFA you produced in the previous step (4 points)



Part 3: Grammars (10 points)

Let G be the grammar:

$$
\begin{aligned}
S &\rightarrow ABC \\
A &\rightarrow xB \mid \lambda \\
B &\rightarrow yC \\
C &\rightarrow Az
\end{aligned}
$$

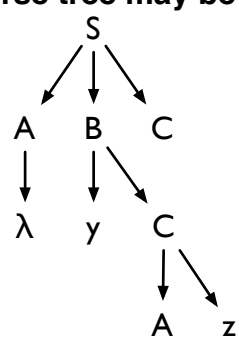
Using this grammar, answer the following questions.

1) What are the terminals and non-terminals of this grammar? (1 point)
Terminals: {x, y, z} Non-terminals: {S, A, B, C}

2) Give 4 examples of strings in the language defined by this grammar. (2 points)

Many possibilities. Note that if there are *n* xs in the string, there must be *n+1* ys, and *n +2* zs.

3) Draw the parse tree for the following partial derivation (*i.e.*, some of the leaves of your parse tree may be non-terminals) (4 points)



$$S \Rightarrow^* yAzC$$

4) Did this partial derivation get produced by left-derivation or right-derivation? (1 points)

Left

5) Give an example of a partial derivation produced after 2 steps of *right*-derivation (2 points)

S ⇒ ABC ⇒ ABAz

Part 4: LL parsers (22 points)

Answer the questions in this part using the following grammar:

$$
\begin{array}{lcl}
S & \rightarrow & Ax\$ \\
A & \rightarrow & yz \\
A & \rightarrow & zAA \\
A & \rightarrow & \lambda
\end{array}
$$

1) Define the following sets: (8 points)

First(Ax$ *)*	{y, z, λ}
First (yz *)*	y
First (zAA *)*	z
Follow (A *)*	{x, y, z}

2) Give the predict sets for the productions: (8 points)

Predict(1)	{x, y, z}
Predict(2)	{y}
Predict(3)	{z}
Predict(4)	{x, y, z}

3) Fill in the LL(1) parse table based on your predict sets (4 points)

	x	y	z	$
S	1	1	1	
A	4	2, 4	3, 4	

4) Is this an LL(1) grammar? Why or why not? (2 points)

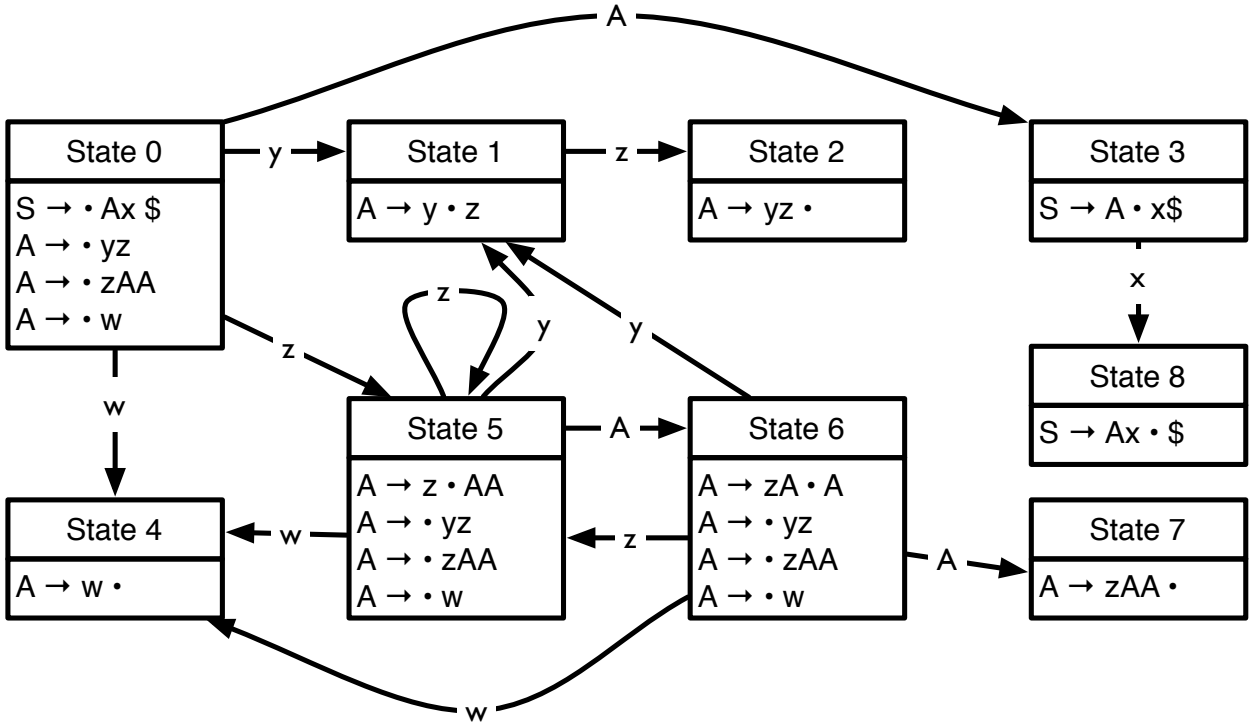
No, because there are conflicts in the parse table.

Part 5: LR(0) Parsers (40 points)

Use the following grammar for the next two questions:

$$
\begin{aligned}
S &\rightarrow Ax\$ \\
A &\rightarrow yz \\
A &\rightarrow zAA \\
A &\rightarrow w
\end{aligned}
$$

1) Fill in the missing states for the for the following CFSM (22 points) and fill in the missing edge labels (10 points)



2) List the actions the parser will take when parsing the following string. For shift actions, indicate which state the parser will go to; for reduce actions, indicate which rule is being reduced and which state the parser will go to after reducing. You do not have to show the parse stack or the remaining input—though it may help. Assume the parser accepts when it gets to state 8. (8 points)

zwwx $

Shift 5, Shift 4, Reduce 4 (goto 6), Shift 4, Reduce 4 (goto 7), Reduce 3 (goto 3), Shift 8, Accept