CS 164, Fall/2000
Midterm #1
Professor Aiken

Problem #1 - Regular Expressions and Finite Automata (15 points)

Consider the design of a small language using only the letters "z", "o", and the slash character "/". A comment in this language starts with "/o" and ends after the very next "o/". Comments do not nest.

Give a single regular expression that matches exactly one complete comment and nothing else. For full credit, use only the core regular expression notations *A+B, AB, A*, A+,* *∉epsilon),* and *"abc"*.

Problem #2 - Semantic Actions (25 points)

Consider the following grammars and associated semantic actions. In the actions, the operations *And, Or,* and *Not* are constructors for an abstract syntax tree data type. For each grammar, answer three things.

- Say whether each attribute of a non-terminal is inherited or synthesized and why.
- Show the value of the attributes of G after parsing ¬(A^(A=>B)).

(a)

$G \rightarrow F$	$G.p = F.p$
$F \rightarrow F_1 \wedge F_2$	$F.p = And(F_1.p, F_2.p)$
$F \rightarrow F_1 \vee F_2$	$F.p = Or(F_1.p, F_2.p)$
$F \rightarrow \neg F_1$	$F.p = Neg(F_1.p)$
$F \rightarrow F_1 \Rightarrow F_2$	$F.p = Or(Not(F_1.p), F_2.p)$
$F \rightarrow (F_1)$	$F.p = F_1.p$
$F \rightarrow id$	$F.p = id.lexeme$

(b) Remember: Say whether each attribute of a non-terminal is inherited or synthesized and why. Show the value of the attributes of G after parsing ¬(A^(A=>B)).

$G \rightarrow F$	$G.q = F.q$
	$F.b = true$
$F \rightarrow F_1 \wedge F_2$	$F.q = F.b \,?\, And(F_1.q, F_2.q) : Or(F_1.q, F_2.q)$
	$F_1.b = F.b$
	$F_2.b = F.b$
$F \rightarrow F_1 \vee F_2$	$F.q = F.b \,?\, Or(F_1.q, F_2.q) : And(F_1.q, F_2.q)$
	$F_1.b = F.b$
	$F_2.b = F.b$
$F \rightarrow \neg F_1$	$F.q = F_1.q$
	$F_1.b = \neg F.b$
$F \rightarrow F_1 \Rightarrow F_2$	$F.q = F.b \,?\, Or(F_1.q, F_2.q) : And(F_1.q, F_2.q)$
	$F_1.b = \neg F.b$
	$F_2.b = F.b$
$F \rightarrow (F_1)$	$F.q = F_1.q$
	$F_1.b = F.b$
$F \rightarrow id$	$F.q = F.b \,?\, id.lexeme : Neg(id.lexeme)$

Problem #3 - First and Follow Sets (20 points)

Give a grammar with the following **First** and **Follow** sets. Your grammar should have exactly two productions per non-terminal and no epsilon productions. The non-terminals are *X, Y, Z* and the terminals are *a, b, c, d, e, f.*

First(X)	=	$\{b, d, f\}$	Follow(X)	=	$\{\$\}$
First(Y)	=	$\{b, d\}$	Follow(Y)	=	$\{c, e\}$
First(Z)	=	$\{c, e\}$	Follow(Z)	=	$\{a\}$
Follow(d)	=	$\{c, e\}$	Follow(a)	=	$\{\$\}$
Follow(e)	=	$\{a\}$	Follow(b)	=	$\{b, d\}$
Follow(f)	=	$\{\$\}$	Follow(c)	=	$\{c, e\}$

Problem #4 - LR Parsing and Error Recovery (25 points)

Consider the following grammar.

```
S -> S a |b| error a
```

(a) Show the DFA for recognizing viable prefixes of this grammar. Use LR(0) items, and treat `error` as a terminal.
(b) Tools such as `bison` use error productions in the following way. When a parsing error is encountered (i.e., the machine cannot shift, reduce, or accept):

- First, pop and discard elements of the stack one at a time until a stack configuration is reached where the `error` terminal of an error production can be shifted on to the stack.
- Second, discard tokens of the input one at a time until one is found that can be shifted on to the stack.
- Third, resume normal execution of the parser.

Show the sequence of stack configurations of an SLR(1) parser for the grammar above on the following input. Be sure to show all shift, reduce, and discard actions (for both stack and input).

```
bacadfa
```

Problem #5 - Miscellaneous Parsing (15 points)

(a) Give an unambiguous grammar that is not LR(1).

(b) How many strings are in the language of the grammar `S -> aS`?

(c) Which of LL(1), SLR(1), and LR(1) can parse strings in the following grammar, and why?

```
E -> A|B
A -> a|c
B -> b|c
```