CS143 Midterm
Spring 2014

- Please read all instructions (including these) carefully.
- There are 5 questions on the exam, most with multiple parts. You have 75 minutes to work on the exam.
- The exam is open note.
- Please write your answers in the space provided on the exam, and clearly mark your solutions. You may use the backs of the exam pages as scratch paper. Please do not use any additional scratch paper.
- Solutions will be graded on correctness and clarity. Each problem has a relatively simple and straightforward solution. You may get as few as 0 points for a question if your solution is far more complicated than necessary. Partial solutions will be graded for partial credit.

NAME: ____

In accordance with both the letter and spirit of the Honor Code, I have neither given nor received assistance on this examination.

SIGNATURE: ____

Problem	Max points	Points
1	10	
2	15	
3	15	
4	20	
5	15	
TOTAL	75	

1. **Lexical Analysis** (10 points)
   Consider the following flex-like lexical specification:

```
a*b   { print "1" }
ca    { print "2" }
a*ca* { print "3" }
```

   Given the following input string

   abcaacacaaabbaaabcaaca

   what does the lexer print?

   132311132

2. **Top-Down Parsing** (15 points)

One of your coworkers at PGP (Pretty Good Parsing) spills coffee on her LL(1) parsing table, obliterating the row labels. Besides the surviving portion of the table, she remembers the following facts about the grammar:

$2 \in \mathit{Follow}(Z)$
$\epsilon \notin \mathit{First}(M)$
$\epsilon \notin \mathit{First}(J)$
$\mathit{First}(M) \subseteq \mathit{First}(K)$.

Label each row of the table with one of the non-terminals M, K, Z, J or C such that the parsing table is consistent and the facts above are satisfied.

	1	2	3	4	5	\$
K	MC				MC	
C		$2K$		ϵ		ϵ
M	JZ				$5Z$	
J	$1K4$					
Z		ϵ	$3M$	ϵ		ϵ

3. **Grammars** (15 points)

- Give an example of a grammar that generates an infinite language where there is a unique derivation for every parse tree. For full credit, give a grammar that is as simple as possible.

```
S -> aS
S -> epsilon
```

- State a characteristic property of every grammar where every parse tree has a unique derivation.

```
Every production has at at most one non-terminal on the right-hand side.
```

- Can a grammar where every parse tree has a unique derivation be ambiguous? If the answer is yes, give an example grammar. If the answer is no, explain why not.

```
S -> X
S -> Y
X -> a
Y -> a
```

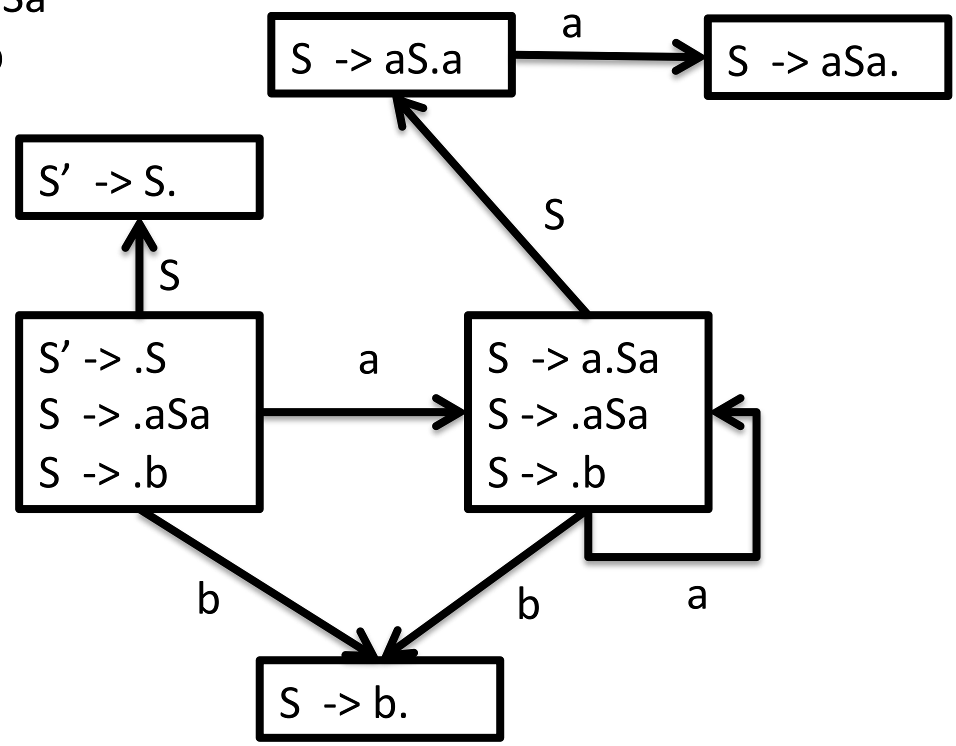
4. **Bottom-Up Parsing** (20 points)
Below is the skeleton of an LR parsing automaton, showing just the states and transitions.

- Write a grammar that has this state diagram.

- Fill in the states with the LR(0) sets of items from your grammar and label the edges with the correct transition symbol. There should be no shift-reduce or reduce-reduce conflicts in your automaton (under LR(0) parsing rules). For full credit, use the minimum number of terminals and non-terminals.

Grammar:

S’ -> S
S -> aSa
S -> b



5. **Syntax-Directed Translation** (15 points)

Complete the actions for the following syntax-directed translation. A program P is a sequence of statements separated by semicolons. Each statement assigns the value of an expression E to the variable x. An expression is either the sum of two expressions, the constant 1, or the current value of x.

Statements are evaluated in left-to-right order. For the ith statement $x = E_i$, the value of references to x inside E_i is the value assigned to x in the previous statement $x = E_{i-1}$. For the first statement $x = E_1$ the value of references to x in E_1 is 0. The value of a program is the value assigned to x by the last statement.

Each action should be a set of equalities of the form

$$a_0 = F(a_1, \ldots, a_n)$$

where F is some function of the attributes a_i. Assume the parser will figure out the order in which to solve the equations, computing the right-hand side before assigning it to the attribute named on the left-hand side. Do not use anything but attribute equations and do not use any global state.

Your solution should assign $P.val$ the value of the program generated by P. Introduce whatever other attributes you need to compute the value of a program; indicate for each attribute whether it is inherited or synthesized.

x is an inherited attribute
val is a synthesized attribute

$P_0 \rightarrow P_1; x = E \quad \{E.x = P_1.val \quad P_0.val = E.val\}$

$P \rightarrow \epsilon \quad \{P.val = 0\}$

$E \rightarrow x \quad \{E.val = E.x\}$

$E \rightarrow 1 \quad \{E.val = 1\}$

$E_0 \rightarrow E_1 + E_2 \quad \{E_0.val = E_1.val + E_2.val \quad E_1.x = E_0.x \quad E_2.x = E_0.x\}$