Midterm I Solutions

- Please read all instructions (including these) carefully.
- There are six questions on the exam, some with multiple parts. You have 1:20 to work on the exam.
- The exam is closed book, but you may refer to your four sheets of prepared notes.
- 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: ________________________

SID or SS#: ________________________

Circle the time of your section: 9:00 10:00 11:00 12:00

1:00 2:00 3:00 4:00

Problem	Max points	Points
1	15	
2	15	
3	25	
4	10	
5	25	
6	10	
TOTAL	100	

1. **First and Follow Sets** (15 points)

Consider the following three grammars

1.	$A \rightarrow BC$	***First(A)*** $=$ **{ *x* }**	***Follow(A)*** $=$ **{ *x*, \$ }**
	$B \rightarrow Ax \mid x$	***First(B)*** $=$ **{ *x* }**	***Follow(B)*** $=$ **{ *y* }**
	$C \rightarrow yC \mid y$	***First(C)*** $=$ **{ *y* }**	***Follow(C)*** $=$ **{ *x*, \$ }**
2.	$A \rightarrow BC$	***First(A)*** $=$ **{ *x*, *y* }**	***Follow(A)*** $=$ **{ *x*, \$ }**
	$B \rightarrow Ax \mid x \mid \epsilon$	***First(B)*** $=$ **{ *x*, *y*, ϵ }**	***Follow(B)*** $=$ **{ *y* }**
	$C \rightarrow yC \mid y$	***First(C)*** $=$ **{ *y* }**	***Follow(C)*** $=$ **{ *x*, \$ }**
3.	$A \rightarrow BC$	***First(A)*** $=$ **{ *x*, *y*, ϵ }**	***Follow(A)*** $=$ **{ *x*, \$ }**
	$B \rightarrow Ax \mid x \mid \epsilon$	***First(B)*** $=$ **{ *x*, *y*, ϵ }**	***Follow(B)*** $=$ **{ *x*, *y*, \$ }**
	$C \rightarrow yC \mid y \mid \epsilon$	***First(C)*** $=$ **{ *y*, ϵ }**	***Follow(C)*** $=$ **{ *x*, \$ }**

For each of the following statements about First and Follow sets, indicate for which grammar(s) (1, 2, or 3) the statement is correct. Each statement is correct for one or more grammars—be sure to list all the grammars for which a statement is correct.

Note that grammar 3 is a small extension of grammar 2, which is a small extension of grammar 1. You may find it helpful to compute the First and Follow sets beginning with the simplest grammar.

- $First(A) = \{x, y\} =$ **{ 2 }**
- $Follow(A) = \{\$, x\} =$ **{ 1, 2, 3 }**
- $Follow(B) = \{\$, x, y\} =$ **{ 3 }**
- $First(C) = \{y\} =$ **{ 1, 2 }**
- $Follow(C) = \{\$, x\} =$ **{ 1, 2, 3 }**

2. **Regular Expressions and Context-Free Grammars**(15 points)

In this problem we will show that every regular language is also a context-free language. The proof is to construct for every regular expression an equivalent context-free grammar. A regular expression is equivalent to a context-free grammar if they generate the same language.

We will do the first case for you; the problem is to show how to do two other cases. Consider the regular expression AB. Assume that we have already converted A to an equivalent context-free grammar with start symbol S_A and that we have converted B to an equivalent context-free grammar with start symbol S_B. Then a context-free grammar for the language AB with start symbol S_{AB} is

$$S_{AB} \rightarrow S_A S_B$$

(We also assume that the grammars for S_A and S_B share no non-terminals.)

(a) In the same style as above, what is a context-free grammar for $A + B$?

Assume that we have already converted A to an equivalent context-free grammar with start symbol $\mathtt{S_A}$ and that we have converted B to an equivalent context-free grammar with start symbol $\mathtt{S_B}$. Then a context free grammar for $\mathtt{A+B}$ with start symbol $\mathtt{S_{A+B}}$ is

$$\mathtt{S_{A+B}} \rightarrow \mathtt{S_A} \mid \mathtt{S_B}$$

(b) What is a context-free grammar for A^*?

Assume that we have already converted A to an equivalent context-free grammar with start symbol $\mathtt{S_A}$. Then a context free grammar for $\mathtt{A^*}$ with start symbol $\mathtt{S_{A^*}}$ is

$$\mathtt{S_{A^*}} \rightarrow \mathtt{S_A S_{A^*}} \mid \varepsilon$$

3. **LR Parsing** (25 points)

In these problems we describe a DFA for accepting viable prefixes of a grammar. Your assignment is to give a grammar whose DFA recognizing viable prefixes satisfies the description.

Assume the DFA's we describe contain LR(0) items—no lookaheads are involved. Do not augment your grammar with an extra production $S \to S'$. Just find any grammar that works.

Many people misunderstood this problem. Reread the first paragraph carefully. Instead of giving a grammar whose DFA for recognizing viable prefixes matches the given description, many people gave a grammar that generates a (regular) language that has a DFA matching the description. Notice that these answers don't really make sense, in any case, since there are many possible DFAs for the same language. It is a coincidence that the languages generated by the grammars for parts (a) and (b) happen to have DFAs that match the descriptions.

(a) The DFA consists of one state and no transitions.

$$\mathsf{S} \to \varepsilon$$

The state contains the item $\mathsf{S} \to .$

(b) The DFA consists of two states s_1 and s_2 and there is one transition from the start state s_1 to s_2.

$$\mathsf{S} \to \mathsf{a}$$

State s_1 contains the item $\mathsf{S} \to .\mathsf{a}$ and state s_2 contains the item $\mathsf{S} \to \mathsf{a}.$

(c) The DFA consists of three states s_1, s_2, and s_3. There is one transition from the start state s_1 to s_2, one transition from s_2 to itself, and one transition from s_2 to s_3.

There were several similar acceptable answers to this problem.

$$\mathsf{S} \to \mathsf{aS}$$

$$\mathsf{S} \to \mathsf{aS} \mid \varepsilon$$

$$\mathsf{S} \to \mathsf{aS} \mid \mathsf{a}$$

The first grammar generates the empty language (it generates no finite strings). For the first grammar, the states are

$$s_1: \ \mathsf{S} \to .\mathsf{aS} \qquad s_2: \ \begin{array}{l} \mathsf{S} \to \mathsf{a}.\mathsf{S} \\ \mathsf{S} \to .\mathsf{aS} \end{array} \qquad s_3: \ \mathsf{S} \to \mathsf{aS}.$$

The second grammar generates the language containing all strings of a's, and the third generates all strings of a's of length at least 1.

4. **Syntax-Directed Translation** (10 points)

For the following grammar for arithmetic expressions define a syntax-directed translation that records the sign (POS or NEG) of each subexpression E in the attribute $E.sign$. Write your semantic actions to the right of the corresponding production. You may assume that all integers are non-zero (so you don't need to worry about the sign of zero).

$E \rightarrow$ *unsigned_integer*
{ E.sign = POS; }

$\mid$ $+E_1$
{ E.sign = E_1.sign; }

$\mid$ $-E_1$
{ E.sign = (E_1.sign == POS) ? NEG : POS; }

$\mid$ $E_1 * E_2$
{ E.sign = (E_1.sign == E_2.sign) ? POS : NEG; }

Common mistakes in answers to this problem were:

- Not realizing that a sign of an unsigned integer is always POS.
- Assuming that +E and -E force the sign of the result to be POS and NEG respectively (the grammar *is* for arithmetic expressions, after all).
- Using operations on POS and NEG without defining them.
- Getting the sign logic in $E_1 * E_2$ wrong.

5. **LL Parsing** (25 points)

Consider the following grammar:

$$\begin{aligned} S &\rightarrow aS \mid Ab \\ A &\rightarrow XYZ \mid \epsilon \\ X &\rightarrow cS \mid \epsilon \\ Y &\rightarrow dS \mid \epsilon \\ Z &\rightarrow eS \end{aligned}$$

(a) Give an LL(1) parsing table for this grammar.

`Even though it was not required in this exercise it is best to first compute the` *First* `and` *Follow* `sets for the non-terminals:`

	First	Follow
S	$\{a, b, c, d, e\}$	$\{\$, b, d, e\}$
A	$\{c, d, e, \epsilon\}$	$\{b\}$
X	$\{c, \epsilon\}$	$\{d, e\}$
Y	$\{d, \epsilon\}$	$\{e\}$
Z	$\{e\}$	$\{b\}$

`Based on this information we can write the LL(1) parsing table as follows:`

	a	b	c	d	e	\$
S	$S \rightarrow aS$	$S \rightarrow Ab$	$S \rightarrow Ab$	$S \rightarrow Ab$	$S \rightarrow Ab$	
A		$A \rightarrow \epsilon$	$A \rightarrow XYZ$	$A \rightarrow XYZ$	$A \rightarrow XYZ$	
X			$X \rightarrow cS$	$X \rightarrow \epsilon$	$X \rightarrow \epsilon$	
Y				$Y \rightarrow dS$	$Y \rightarrow \epsilon$	
Z					$Z \rightarrow eS$	

`The` $X \rightarrow \epsilon$ `entries are put in the columns corresponding to the` $Follow(X)$`.`
`Same for the` ϵ `productions of` A `and` Y`.`
`Somewhat tricky were the entries for` $A \rightarrow XYZ$ `that are placed in the columns in` $First(XYZ) = \{c, d, e\}$ `and the entries for` $S \rightarrow Ab$ `that are placed in the columns in` $First(Ab) = \{b, c, d, e\}$`.`
`This grammar is LL(1) since all the entries in the table are uniquely defined.`

(b) Give a leftmost derivation of the string *aebb*.

A leftmost derivation of the string *aebb* is

$$S \to aS \to aAb \to aXYZb \to aYZb \to aZb \to aeSb \to aeAbb \to aebb$$

Since the grammar is LL(1) (no conflicts in the above parsing table) the grammar is not ambiguous and this leftmost derivation is unique.

(c) Show that if we add the production $X \to bS$ that the grammar is no longer LL(1).

If we add the production $X \to bS$ then the following *First* and *Follow* sets change:

$$\begin{aligned} First(X) &= \{b, c, \epsilon\} \\ First(A) &= \{b, c, d, e, \epsilon\} \end{aligned}$$

Since $b \in First(X)$ we have that $b \in First(XYZ)$ and we will therefore have a conflict in the LL(1) parsing table. The entry at $[A, b]$ will contain both entries $A \to \epsilon$ and $A \to XYZ$. This makes the grammar not LL(1).

Note that the grammar is still unambiguous!

6. **Error Recovery** (10 points)

   Consider the following grammar for strings of ab's:

$$A \to abA \mid \epsilon$$

   (a) Augment the grammar with productions that allow a parser to recognize all erroneous strings as well as valid strings. The non-terminal ERROR should derive the suffix starting at the first erroneous terminal. More precisely, in the case that the input is erroneous, then ERROR should

   - derive a suffix s of the string such that if s is deleted, then the string is in the original language, and
   - derive the shortest such suffix.

   You may use as many new productions and non-terminals besides ERROR as you like. Assume that a and b are the only possible terminals.

$$\begin{array}{rcl} \texttt{A} & \to & \texttt{abA} \mid \epsilon \mid \texttt{ERROR} \\ \texttt{ERROR} & \to & \texttt{a} \mid \texttt{bX} \mid \texttt{aaX} \\ \texttt{X} & \to & \texttt{aX} \mid \texttt{bX} \mid \epsilon \end{array}$$

   (b) What language does your augmented grammar generate?

   `The augmented grammar generates` $(a+b)^*$.

   `Common mistakes in answers to this problem were:`

   - `changing the existing productions;`
   - `treating ERROR as a terminal rather than a non-terminal;`
   - `deriving a prefix rather than a suffix of the string;`
   - `writing productions for ERROR that didn't cover all possible erroneous suffixes.`