Name: ________________________________

<table>
<thead>
<tr>
<th>Number</th>
<th>Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Use only one side of the paper and start each problem on a new page!!

Please show all work to receive ANY credit!!!

1. (25 points) Consider the grammar $G$ shown below, where $S$ is the start symbol:

   $S \longrightarrow AA | \varepsilon$
   $A \longrightarrow SB | x$
   $B \longrightarrow Ay | y$

   a. (6 points) Find all strings in $L(G)$ that can be produced by derivations of 5 or fewer steps. A string is in $L(G)$ if it consists of only terminal symbols.

   b. (4 points) Is $xyxy$ a string in the language? Why or why not?

   c. (15 points) Redesign the grammar given above into an equivalent form that is suitable for top down parsing by first removing any left recursion and then removing any $\varepsilon$ moves.

2. (25 points) The grammar:

   $S \longrightarrow SS | (S) | \varepsilon$

has a cycle, since the derivation

   $S \longrightarrow SS \longrightarrow S$

   can occur. Develop an algorithm that when applied to a grammar will remove cycles. Demonstrate that your algorithm works by applying it to the above grammar. Hint: It may be helpful for you to consider the strings being generated by the grammar to answer this question.
3. (20 points) Consider the grammar, with FIRST and FOLLOW computed for each non-terminal in the grammar:

1. A ---> BA'   \quad \text{FIRST}(A) = \{ x, ( \}   \quad \text{FOLLOW}(A) = \{ \$ \}   \\
2. A' ---> oBA' \quad \text{FIRST}(A') = \{ o, e \}   \quad \text{FOLLOW}(A') = \{ \$ \}   \\
3. A' ---> \epsilon \quad \text{FIRST}(A') = \{ x, ( \}   \quad \text{FOLLOW}(A') = \{ \$ \}   \\
4. B ---> D \quad \text{FIRST}(B) = \{ x, ( \}   \quad \text{FOLLOW}(B) = \{ \$ , o \}   \\
5. B ---> (C) \quad \text{FIRST}(C) = \{ x \}   \quad \text{FOLLOW}(C) = \{ \}   \\
6. C ---> DC' \quad \text{FIRST}(C') = \{ a, e \}   \quad \text{FOLLOW}(C') = \{ \}   \\
7. C' --->  aDC' \quad \text{FIRST}(D) = \{ x \}   \quad \text{FOLLOW}(D) = \{ \$ , o, a \}   \\
8. C' ---> \epsilon   \\
9. D ---> x

a. (18 points) As part of the design of a top-down parser, construct the LL(1) parsing table for the grammar. Use the rule numbers to identify your table entries.

b. (2 points) Is the grammar LL(1)? Why or why not?

Selected Answers for Exam 1 - CS244 - Spring 1992 Semester

#1a. Strings in L(G) are: epsilon, xx, xy, and yx

#1b. S ---a AA ---> SBA ---> AABA ---> SBABA ---> BABA ---> yABA
     ---> yxBA ---> yxyA ---> yxySB ---> yxyB ---> yxyAy ---> yxyxy

#1c. S ---> AA | epsilon
    A ---> BA' | xA' | B | x
    A' ---> ABA' | AB
    B ---> xA'yB' | yB' | xyB' | xA'y | xy | y
    B' ---> A'yB' | yB' | A'y | y

#2. TRY IT AGAIN!!! (SEE NEXT PAGE)

#3. Where would the sync actions go?

<table>
<thead>
<tr>
<th>a</th>
<th>o</th>
<th>x</th>
<th>(</th>
<th>)</th>
<th>$</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A'</td>
<td>2</td>
<td></td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>4</td>
<td></td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td></td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C'</td>
<td>7</td>
<td></td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td></td>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b. No multiply defined entries
#2 \[ S \rightarrow SS \mid (S) \mid \epsilon \]

**STEP 1: REMOVE LR.**

\[ S \rightarrow (S)s' \mid s' \]

\[ s' \rightarrow SS' \mid \epsilon \]

**STEP 2: REMOVE E-MOVES**

\[ S \rightarrow (S)s' \mid s' \mid (S) \mid \epsilon \] **CYCLE STILL POSSIBLE**

\[ s' \rightarrow SS' \mid (S) \]

\[ S \Rightarrow S' \Rightarrow S \]

**STEP 3: REMOVE E-MOVES**

\[ S \rightarrow (S)s' \mid (S) \mid (S)s' \mid (S) \]

\[ s' \rightarrow s' \mid \epsilon \] **REDUNDANT, SO OMIT!**