Regular Expressions and Finite Automata (20 points)

Write a regular definition for a 32-bit IP address, which consists of four numbers between 0-255, separated by dots. (4 points)

Convert this regular expression into an NFA. Label each state with a number. (8 points)

\[(\text{moo}^* | \text{m}^* | \text{milk})^*\]

Convert the NFA above into an equivalent DFA. Please label each state with an upper case letter, and indicate which NFA states correspond to each DFA state. (8 points)
Top Down Parsing (20 points)

Point out all the ways in which this grammar is not LL(1). (2 points)

E -> E + T
E -> T
T -> id
T -> id( L );
L -> E , L
L -> E

Write a new grammar that is LL(1) and accepts the same language. (4 points)
(Be sure to number each production rule.)

Write out the First and Follow sets for the new grammar. (4 points)
Write out the LL(1) parse table for the new grammar. (10 points)
(There are more cells in the table than you will actually need.)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

( this space intentionally left blank )
Bottom Up Parsing (20 points)

Draw the LR(0) automaton for the following grammar. (16 points)

S -> ID = E  
E -> E + P  
E -> P  
P -> ID  
P -> ID = E

Is this grammar SLR? Explain why. (2 points)

Is this grammar LR(1)? Explain why. (2 points)