EXAM

Please read all instructions, including these, carefully

• There are 7 questions on the exam, with multiple parts. You have 3 hours to work on the exam.
• The exam is open book, open notes.
• Please write your answers in the space provided on the exam and clearly mark your solutions. Write only the final answer on the exam. Scratch paper is available at the front of the room.
• Each problem has a straightforward solution. Solutions will be graded on correctness and clarity. Partial solutions will be given partial credit.

NAME: ____________________________________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Max points</th>
<th>Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
1. (a) Explain why an LL(1) parser cannot handle left-recursive grammars.

(b) The following production, part of a grammar for arithmetic expressions, is immediately left-recursive. Transform it into an equivalent production or equivalent productions that are not left-recursive in order to provide a suitable basis for the implementation of an LL(1) parser.

\[ \text{Exp} \rightarrow \text{Exp} + \text{Prim} \mid \text{Exp} * \text{Prim} \mid \text{Prim} \]

(c) Based on your transformed grammar fragment from question (b) above, write a parsing function for a recursive descent parser for the non-terminal \text{Exp}.
2. (a) Construct the DFA that recognizes the viable prefixes for the following CFG:

\[ S \rightarrow aAb \mid cS \mid B \]
\[ A \rightarrow eAB \mid Af \mid g \]
\[ B \rightarrow d \]

S, A, and B are nonterminal symbols, S is the start symbol, and a, b, c, d, e, f, and g are terminal symbols.

(b) Construct the SLR(1) parse table for the grammar.
(c) Show the steps of the SLR(1) parser for the string `caegdffb`.

(d) Which entries in the parse table would differ if you had instead constructed an LR(1) parser from this grammar, and what would they be?
3. Consider the following grammar:
   
   \[ S \rightarrow aS \mid Ab \]
   \[ A \rightarrow XYZ \mid \varepsilon \]
   \[ X \rightarrow cS \mid \varepsilon \]
   \[ Y \rightarrow dS \mid \varepsilon \]
   \[ Z \rightarrow eS \]

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

   (b) Show the steps of the parser for the string acebb.
Consider the following context free grammar:

\[
\begin{align*}
S & \rightarrow d a A e \\
A & \rightarrow d a A \mid E \\
E & \rightarrow E b P \mid P \\
P & \rightarrow d \mid pAp
\end{align*}
\]

(a) Is the grammar SLR(1)? Why or why not?

(b) Is the grammar LR(1)? Why or why not?
5. Consider the following three grammars:

(1) \[ A \rightarrow BC \]
    \[ B \rightarrow Ax \mid x \]
    \[ C \rightarrow yC \mid y \]

(2) \[ A \rightarrow BC \]
    \[ B \rightarrow Ax \mid x \mid \varepsilon \]
    \[ C \rightarrow yC \mid y \]

(3) \[ A \rightarrow BC \]
    \[ B \rightarrow Ax \mid x \mid \varepsilon \]
    \[ C \rightarrow yC \mid y \mid \varepsilon \]

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.

- First(\(A\)) = \(\{x,y\}\)
- Follow(\(A\)) = \(\{$,x\}\)
- Follow(\(B\)) = \(\{$,x,y\}\)
- First(\(C\)) = \(\{y\}\)
- Follow(\(C\)) = \(\{$,x\}\)
6. For each grammar explain why, or why not, the grammar is LL(1):

(a) \[ S \rightarrow ABBA \]
\[ A \rightarrow a \mid \varepsilon \]
\[ B \rightarrow b \mid \varepsilon \]

(b) \[ S \rightarrow aSc \mid B \]
\[ B \rightarrow bBe \mid C \]
\[ C \rightarrow cCe \mid d \]
7. Consider the following grammar for specifying binary trees (in linearized form):

\[
\text{BinTree} \rightarrow (\text{num} \ \text{BinTree}_1, \text{BinTree}_2) \mid \epsilon
\]

(a) Extend the above grammar by defining a translation scheme such that a depth-first left-to-right traversal of the parse tree with semantic actions will entail checking if the binary tree is ordered. A binary tree is ordered if the values of the numbers of the first subtree (\text{BinTree}_1) are less than the value of the number of the node currently being visited and the values of the numbers of the second subtree (\text{BinTree}_2) are greater than the value of the number of the node currently being visited. For example, \((2 \ (1 \ \text{nil} \ \text{nil}) \ (3 \ \text{nil} \ \text{nil}))\) is ordered but \((1 \ (2 \ \text{nil} \ \text{nil}) \ (3 \ \text{nil} \ \text{nil}))\) is not.

(b) Show the translation of the input string \((2 \ (1 \ \text{nil} \ \text{nil}) \ (3 \ \text{nil} \ \text{nil}))\) by decorating its parse tree. Make sure your tree is drawn clearly.