COMP 412, Fall 2018
Practice Midterm Exam

Disclaimer: THIS IS A PRACTICE EXAM. These questions are taken from past exams. The questions on the actual exam may or may not be similar.

1. **General Knowledge**
   a. What derivation does a bottom-up LR(1) parser construct?
   b. What derivation does a top-down LL(1) parser construct?
   c. Define *ambiguity* in a context-free grammar. Give an example of an ambiguous context-free grammar.
   d. Which is smaller: a parse tree or an abstract syntax tree?

2. **Comparisons**
   In each part of this question, you are given two terms. Briefly define each term, and then explain the distinction or difference between them. “Briefly” means a couple of sentences, not a couple of paragraphs.
   a. Deterministic finite automaton (DFA) versus non-deterministic finite automaton (NFA)
   b. Hopcroft’s algorithm for DFA minimization versus Brzozowski’s algorithm for DFA minimization.
   c. Clean value versus dirty value in a local register allocator
   d. Regular expression versus context-free grammar

3. **RE -> NFA -> DFA -> Minimal DFA**
   Consider the following the regular expression
     \[ a \ b^* ( c \ e \mid d \ e ) \]
   a. Write down, by inspection, a DFA to recognize words described by this regular expression.
   Now, show the steps that an automated scanner generator might take with this same regular expression.
   b. Use Thompson’s construction to build an NFA
   c. Use the subset construction to convert the NFA from question 3.c to a DFA
   d. Use Hopcroft’s DFA minimization algorithm to create a minimal DFA.
4. **LR parsing**

The LR family of parsing techniques are the most general form of table-driven, bottom-up parsers.

a. What causes a shift-reduce conflict? Give an example. How is it discovered?

b. What causes a reduce-reduce conflict? Give an example. How is it discovered?

c. At run-time, an LR(1) parser may discover that the input being parsed contains a syntax error. How does the parser discover the error, in terms of the table, the stack, and the input string?
5. **LL(1) Parsing**

a. Write down the algorithm for the LL(1) skeleton parser. That is, given the LL(1) parse table for a grammar, write down the algorithm that uses the table to parse a sentence.

b. Consider the following expression grammar.

```
\begin{array}{|c|}
\hline
0 & Goal \rightarrow Expr \\
1 & Expr \rightarrow Term Expr' \\
2 & Expr' \rightarrow \pm Term Expr' \\
3 & \ | \ - Term Expr' \\
4 & \ | \ \varepsilon \\
5 & Term \rightarrow Factor Term' \\
6 & Term' \rightarrow \ast Factor Term' \\
7 & \ | \ \lfloor Factor Term' \\
8 & \ | \ \varepsilon \\
9 & Factor \rightarrow \{( Expr \} \\
10 & \ | \ number \\
11 & \ | \ identifier \\
\hline
\end{array}
```

An LL(1) parser generator built the following parse table for this grammar.

<table>
<thead>
<tr>
<th></th>
<th>eof</th>
<th>+</th>
<th>-</th>
<th>*</th>
<th>/</th>
<th>(</th>
<th>)</th>
<th>identifier</th>
<th>number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goal</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Expr</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Expr'</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Term</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Term'</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Factor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

Using the parse table to drive the algorithm that you gave in part (a), show the sequence of actions that the parser takes on the input string “x * (2 + * y)”.

Show the state of the parse tree and the lookahead symbol at each step, including the step when the syntax error is discovered. What is the last action that the parser takes prior to discovering the error?