1. Show top-down, left to right, derivation for: \((i + i) - i * i) / i - i\)
   
   Left to right derivation means to expand non-terminals on the left down to terminals before expanding other non-terminals. Use the unambiguous expression, term, factor grammar (E, T, F grammar).

   \[ E \rightarrow \ldots \]

2. Write a grammar for \(L = \{a^n b^m c^m d^n \mid n \geq 1, m \geq 0\}\). The alphabet is \(\{a, b, c, d\}\).
   
   In this language, the number of \(a\)'s generated are the same as the number of \(d\)'s in a string.
   
   The number of \(b\)'s are the same as the number of \(c\)'s in the string.
   
   Example strings include \(ad\ aadd\ aabbbbccccdd\ abcd\ aaaaabbcddddd\)
   
   Note that \(\lambda\) is not in the language.

3. Write a grammar for balanced parentheses. The alphabet includes only the parentheses.
   
   Example strings include \(\lambda\ ()\ (((()))(())()())()()())((())())\)

4. (a). Using the expression/term/factor grammar, draw the parse tree for
   
   \(((a+a) + a + a * (a+a))\)
   
   (b). Using the SLR parse table (linked from web site), parse the string showing stack, remaining input, and actions, in a table as shown:

<table>
<thead>
<tr>
<th>Stack</th>
<th>Input</th>
<th>Action</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>((a+a) + a + a * (a+a))$</td>
<td>s4</td>
<td></td>
</tr>
<tr>
<td>0 4</td>
<td>(a+a) + a + a * (a+a)$</td>
<td>s4</td>
<td></td>
</tr>
</tbody>
</table>

   and so on

5. Attempt to parse the string using the same table structure as in #4: \(a+a\)