1. A function is said to be computable, or recursive, if it is computed by some machine. There are uncomputable functions, such as the busy beaver function. We write \( \Sigma(n) \) for the \( n \)th value of the busy beaver function, as explained on the following web page.

http://googology.wikia.com/wiki/Busy_beaver_function

Does there exist a machine with no input whose output is the list consisting of the first 100 values of the busy beaver function in order, that is: \( \Sigma(1), \Sigma(2), \ldots \Sigma(100) \)?

2. State the pumping lemma for context-free languages. Be precise, and write legibly.
3. Complete the ACTION and GOTO tables of an LALR parser for the grammar given below. This
grammar ambiguously generates the an algebraic language which has the operators addition, subtraction,
multiplication, and negation. Since the grammar is ambiguous, you must resolve the shift-reduce conflicts
by making choices in the action table. The parsing must be consistent with the precedence rules of C++. That
is, addition, subtraction, and multiplication are left-associative, and multiplication has precedence
over addition and subtraction, and negation has the highest precedence. The token id represents an
arbitrary identifier.

<table>
<thead>
<tr>
<th>id</th>
<th>+</th>
<th>−</th>
<th>*</th>
<th>(</th>
<th>)</th>
<th>eof</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>E</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>halt</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. $E \rightarrow E_{1,11} +_2 E_3$
2. $E \rightarrow E_{1,11} -_4 E_5$
3. $E \rightarrow E_{1,3,5,11} *_6 E_7$
4. $E \rightarrow -_8 E_9$
5. $E \rightarrow (_{10}E_{11})_{12}$
6. $E \rightarrow id_{13}$