Infix → Postfix Conversion Algorithms

1. Manual Algorithm:
   (a) Fully parenthesize the the infix expression (one set of parentheses per operator)
   (b) Replace the right parentheses with their corresponding operators
   (c) Remove the left parentheses

   *Example:* \[ \text{A / (B + C) - D} \]
   
   (a) \(( A / ( B + C ) ) - D \)
   
   (b) \(( A ( B C + / ) D - \)
   
   (c) \( A B C + / D - \)

   The infix expression \( A / (B + C) - D \) is the same as the postfix expression \( A B C + / D - \)

2. Stack-based Pseudocode Algorithm:

   ```
   while there are more symbols to be read
   read the next symbol
   case:
      operand --> output it.
      '(' --> push it on the stack.
      ')' --> pop operators from the stack to the output
          until a '(' is popped; do not output either
          of the parentheses.
      operator --> pop higher- or equal-precedence operators
          from the stack to the output; stop before
          popping a lower-precedence operator or
          a '(' . Push the operator on the stack.
   end case
   end while
   pop the remaining operators from the stack to the output
   ```

   *Example:* \[ \text{A / (B + C) - D} \]

<table>
<thead>
<tr>
<th>Input Symbol</th>
<th>Stack Content</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>nil</td>
<td>A</td>
</tr>
<tr>
<td>/</td>
<td>/</td>
<td>A</td>
</tr>
<tr>
<td>(</td>
<td>/(</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>/(</td>
<td>A B</td>
</tr>
<tr>
<td>+</td>
<td>/(*)</td>
<td>A B</td>
</tr>
<tr>
<td>C</td>
<td>/(*)</td>
<td>A B C</td>
</tr>
<tr>
<td>)</td>
<td>/</td>
<td>A B C +</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>A B C + /</td>
</tr>
<tr>
<td>D</td>
<td>=</td>
<td>A B C + / D</td>
</tr>
<tr>
<td>&lt;eof&gt;</td>
<td>nil</td>
<td>A B C + / D -</td>
</tr>
</tbody>
</table>

   The infix expression \( A / (B + C) - D \) is the same as the postfix expression \( A B C + / D - \)