Using Stacks to Translate from Infix to Postfix (General Case)

The algorithm: (uses one stack for storing operators and parentheses)

**Scan infix expression from left to right**
- if (input is a left parenthesis)
  - push it onto the stack
- else if (input is an operand)
  - write it to output
- else if (input is an operator)
  - do until (stack is empty OR top of stack is a left parenthesis OR top of stack is an operator w/ lower precedence than input)
    - read the stack and write the result to output
    - pop the stack
    - push input onto the stack
- else // input is a right parenthesis (spaces are simply discarded)
  - discard input
  - do until (stack is empty OR top of stack is a left parenthesis)
    - read the stack and write the result to output
    - pop the stack
    - if (stack is empty) // unbalanced parentheses
      - print error message and exit
    - else // top of stack is a left parenthesis
      - pop the stack

**Finish up and check error status**
- do until (stack is empty)
  - if (top of stack is a left parenthesis) // unbalanced parentheses
    - print error message and exit
  - else
    - read the stack and write the result to output
    - pop the stack

E.g.: Use a stack to translate $3 * X + ( Y - 12 ) - Z$ to postfix

**(NOTE: In the following, the stack has the top to the right.)**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Stack</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>3</td>
</tr>
<tr>
<td>X</td>
<td>*</td>
<td>3 X</td>
</tr>
<tr>
<td>+</td>
<td>+</td>
<td>3 X *</td>
</tr>
<tr>
<td>(</td>
<td>+ (</td>
<td>3 X *</td>
</tr>
<tr>
<td>Y</td>
<td>+ (</td>
<td>3 X * Y</td>
</tr>
<tr>
<td>-</td>
<td>+ (</td>
<td>3 X * Y</td>
</tr>
<tr>
<td>12</td>
<td>+ (</td>
<td>3 X * Y</td>
</tr>
<tr>
<td>)</td>
<td>+</td>
<td>3 X * Y</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>3 X * Y</td>
</tr>
<tr>
<td>Z</td>
<td>-</td>
<td>3 X * Y</td>
</tr>
</tbody>
</table>

(Stack is empty: all done)