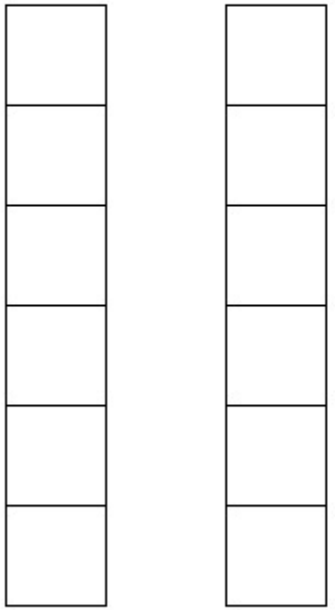
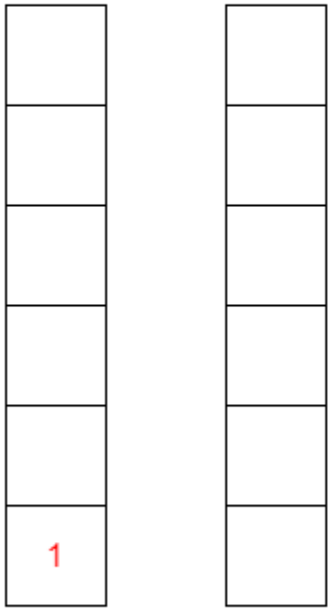
1+2*6-2+(9-6)



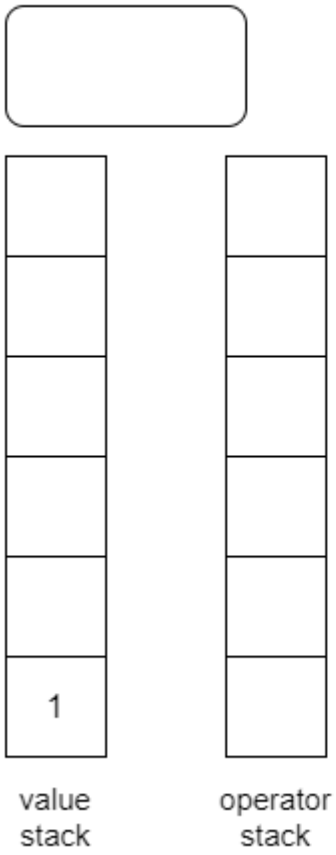
1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit **'0'** to **'9'**: push it onto the **value stack**.
      1.2.2 A left parenthesis **'('**: push it onto the **operator stack**.
      1.2.3 A right parenthesis **')'**:
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis **'('**,
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis **'('** from the **operator stack**, and discard it.
      1.2.4 A **'+'** operator or a **'-'** operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis **'('**,
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the **'+'** or **'-'** onto the **operator stack**.
      1.2.5 A **'*'** operator or a **'/'** operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a **'*'** or a **'/'**),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the **'*'** or **'/'** onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

1+2*6-2+(9-6)



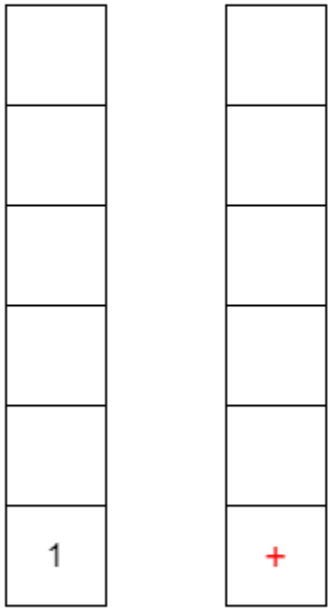
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit **'0'** to **'9'**: push it onto the **value stack**.
    1.2.2 A left parenthesis **'('**: push it onto the **operator stack**.
    1.2.3 A right parenthesis **')'**:
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis **'('**,
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis **'('** from the **operator stack**, and discard it.
    1.2.4 A **'+'** operator or a **'-'** operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis **'('**,
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the **'+'** or **'-'** onto the **operator stack**.
    1.2.5 A **'*'** operator or a **'/'** operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a **'*'** or a **'/'**),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the **'*'** or **'/'** onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



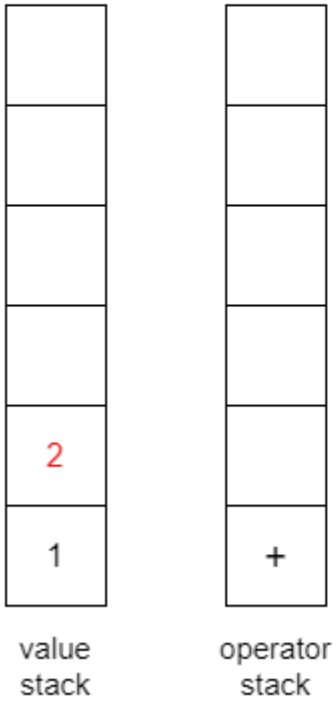
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



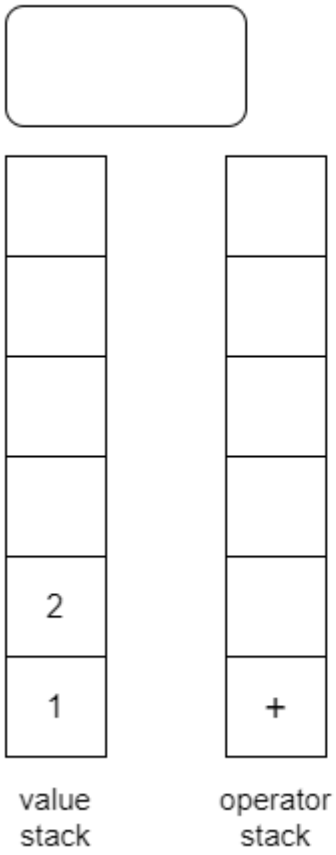
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  Pop it and print the value.

1+2*6-2+(9-6)



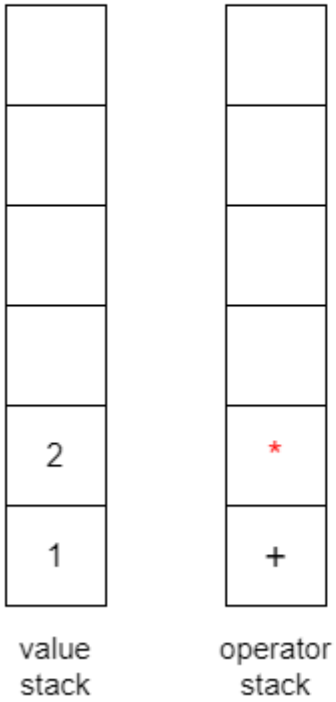
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



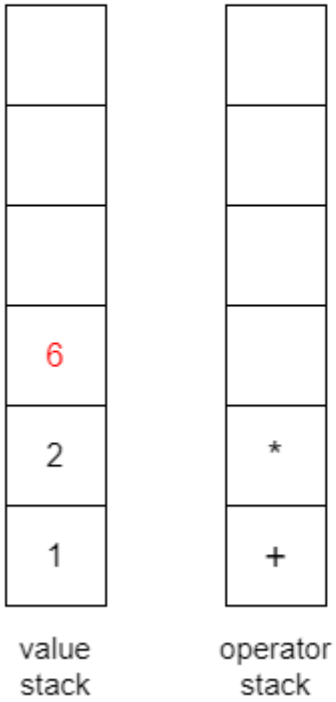
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
      1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
      1.2.3 A right parenthesis '**)**':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
      1.2.4 A '**+**' operator or a '**-**' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
      1.2.5 A '*****' operator or a '**/**' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

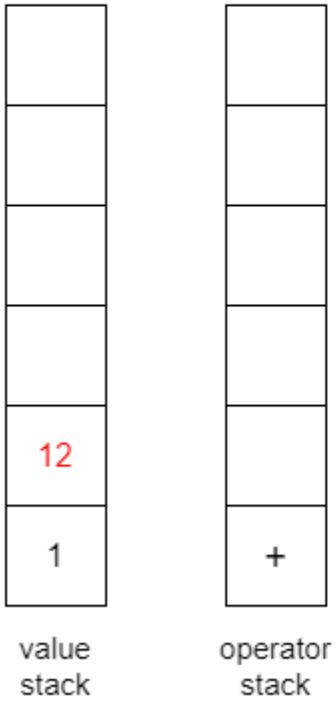
1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '0' to '9': push it onto the **value stack**.
      1.2.2 A left parenthesis '(': push it onto the **operator stack**.
      1.2.3 A right parenthesis ')':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
      1.2.4 A '+' operator or a '-' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '+' or '-' onto the **operator stack**.
      1.2.5 A '*' operator or a '/' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

1+2*6-2+(9-6)

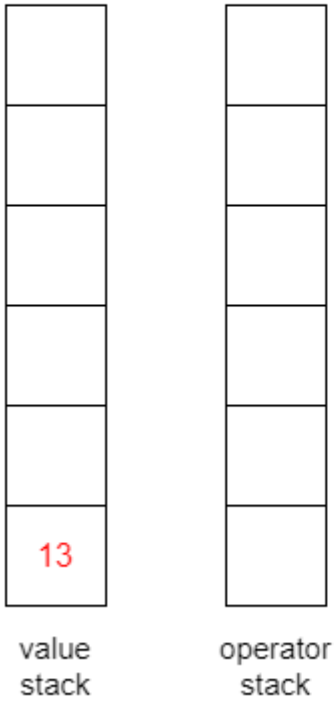
2 * 6 = 12



1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

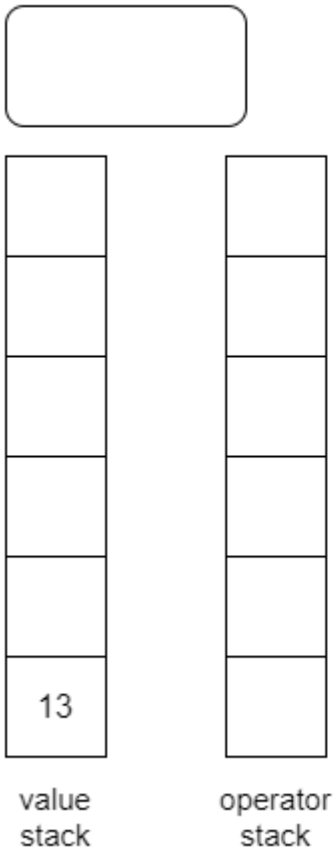
1+2*6-2+(9-6)

1 + 12 = 13



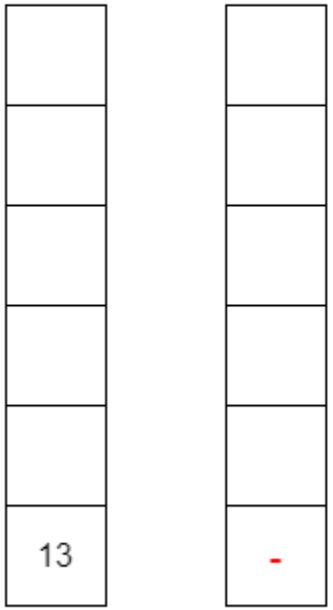
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  Pop it and print the value.

1+2*6-2+(9-6)



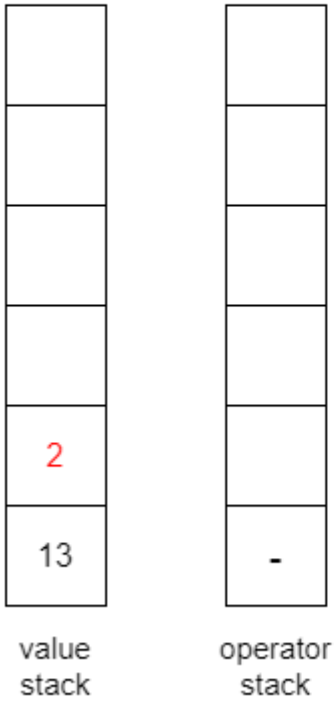
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  Pop it and print the value.

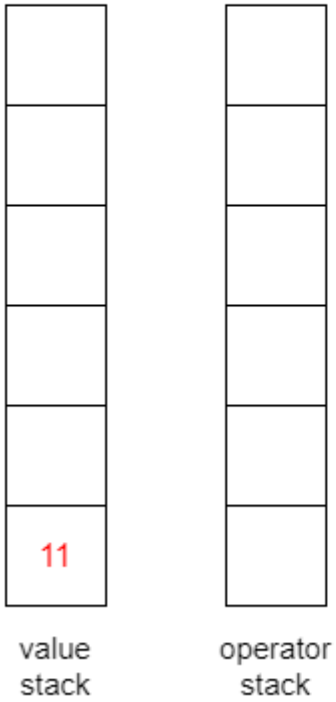
1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '0' to '9': push it onto the **value stack**.
      1.2.2 A left parenthesis '(': push it onto the **operator stack**.
      1.2.3 A right parenthesis ')':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
      1.2.4 A '+' operator or a '-' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '+' or '-' onto the **operator stack**.
      1.2.5 A '*' operator or a '/' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

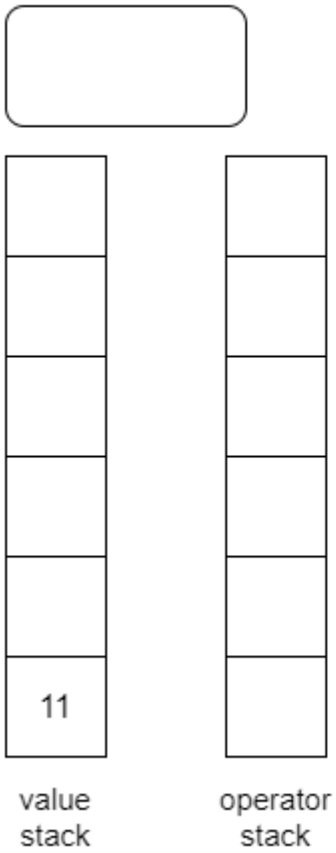
1+2*6-2+(9-6)

13 - 2 = 11



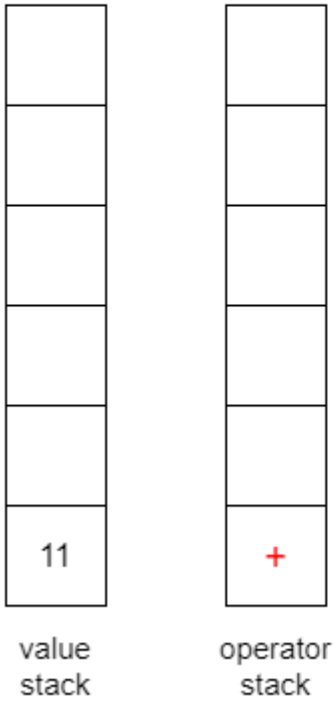
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  Pop it and print the value.

1+2*6-2+(9-6)



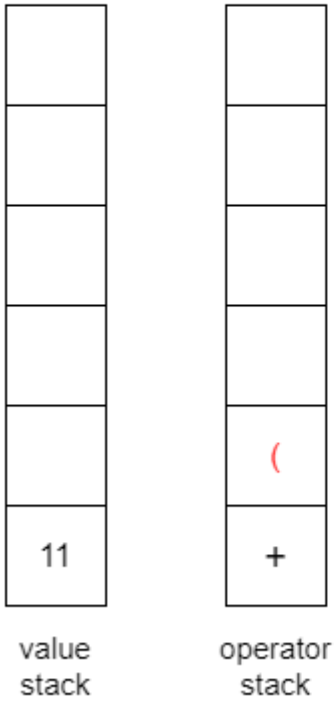
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



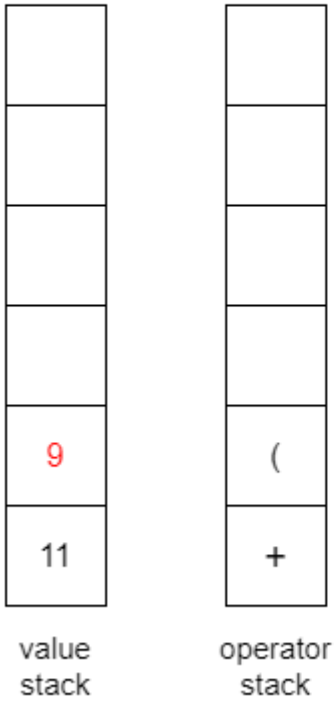
1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '0' to '9': push it onto the **value stack**.
      1.2.2 A left parenthesis '(': push it onto the **operator stack**.
      1.2.3 A right parenthesis ')':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
      1.2.4 A '+' operator or a '-' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '+' or '-' onto the **operator stack**.
      1.2.5 A '*' operator or a '/' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

1+2*6-2+(9-6)



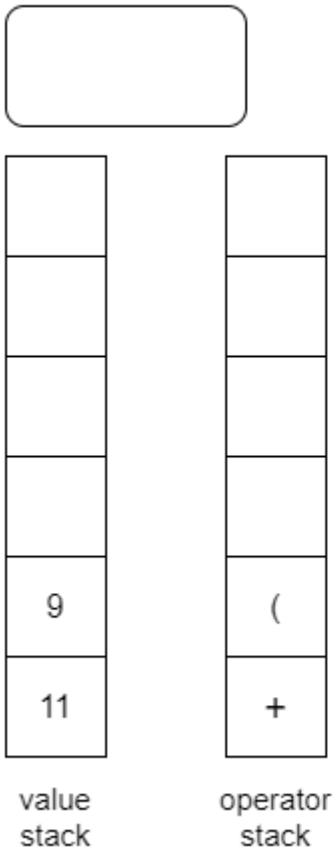
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



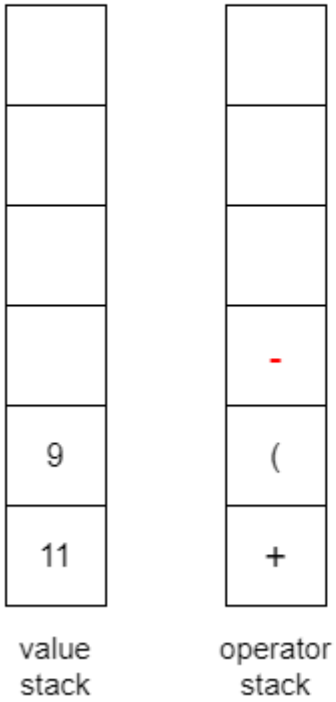
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  Pop it and print the value.

1+2*6-2+(9-6)



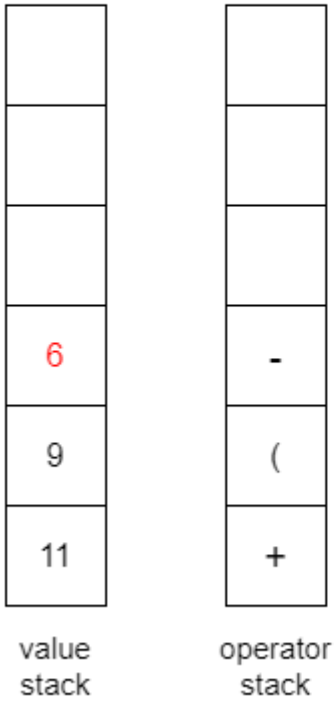
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '0' to '9': push it onto the **value stack**.
    1.2.2 A left parenthesis '(': push it onto the **operator stack**.
    1.2.3 A right parenthesis ')':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
    1.2.4 A '+' operator or a '-' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '+' or '-' onto the **operator stack**.
    1.2.5 A '*' operator or a '/' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  Pop it and print the value.

1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '0' to '9': push it onto the **value stack**.
      1.2.2 A left parenthesis '(': push it onto the **operator stack**.
      1.2.3 A right parenthesis ')':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
      1.2.4 A '+' operator or a '-' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '+' or '-' onto the **operator stack**.
      1.2.5 A '*' operator or a '/' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

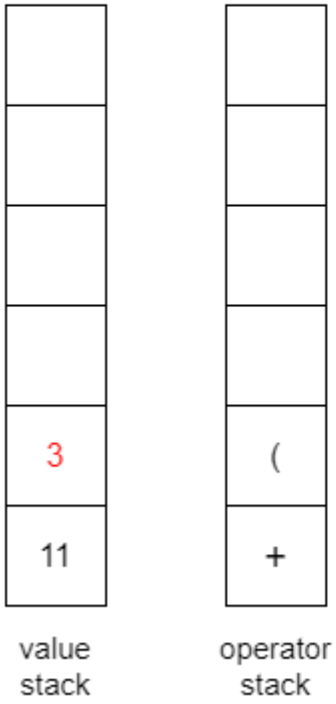
1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '0' to '9': push it onto the **value stack**.
      1.2.2 A left parenthesis '(': push it onto the **operator stack**.
      1.2.3 A right parenthesis ')':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '(',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '(' from the **operator stack**, and discard it.
      1.2.4 A '+' operator or a '-' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '(',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '+' or '-' onto the **operator stack**.
      1.2.5 A '*' operator or a '/' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*' or a '/'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*' or '/' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

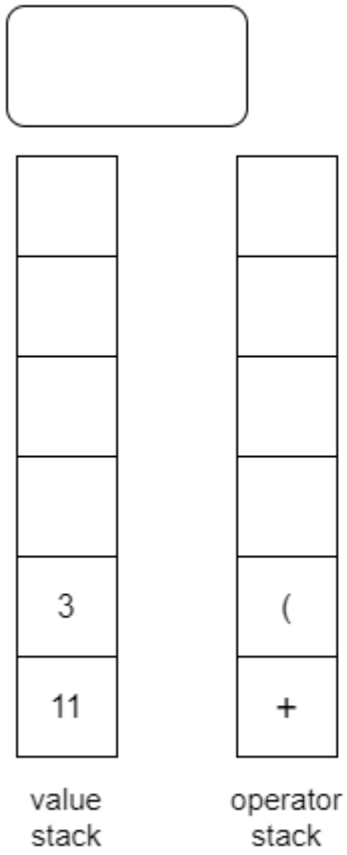
1+2*6-2+(9-6)

9 - 6 = 3



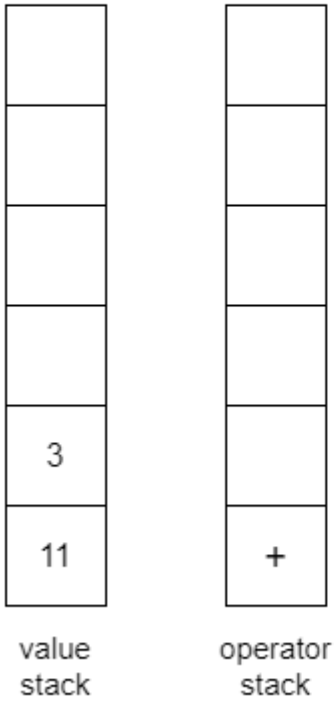
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



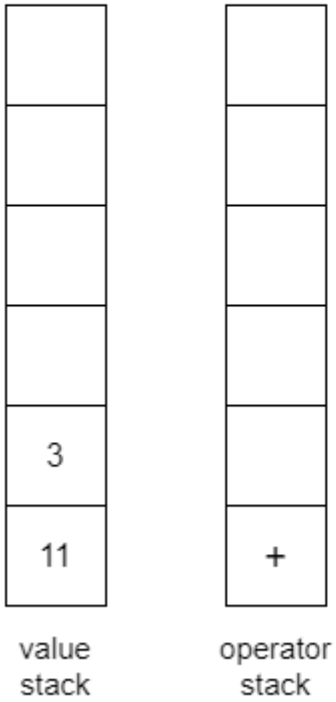
1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)



1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

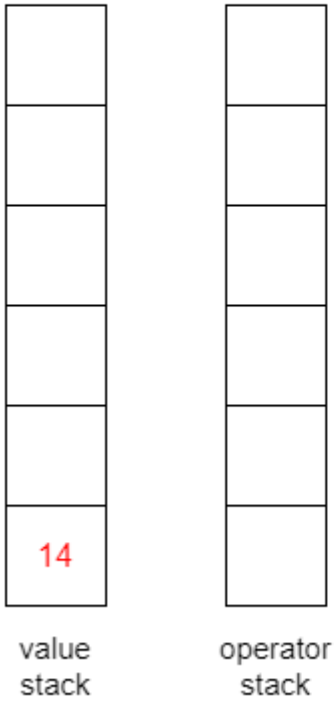
1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
      1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
      1.2.3 A right parenthesis '**)**':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
      1.2.4 A '**+**' operator or a '**-**' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
      1.2.5 A '*****' operator or a '**/**' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

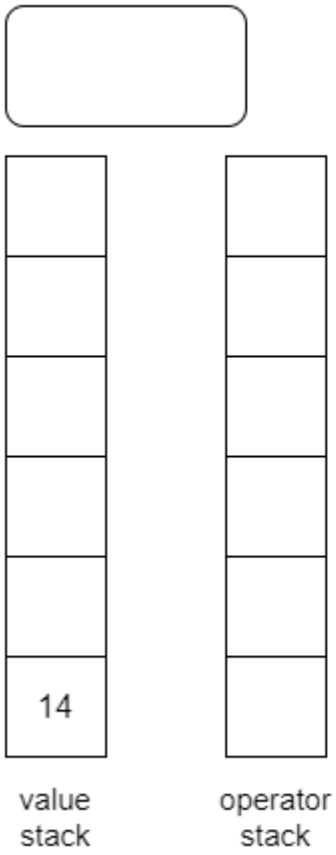
1+2*6-2+(9-6)

11 + 3 = 14



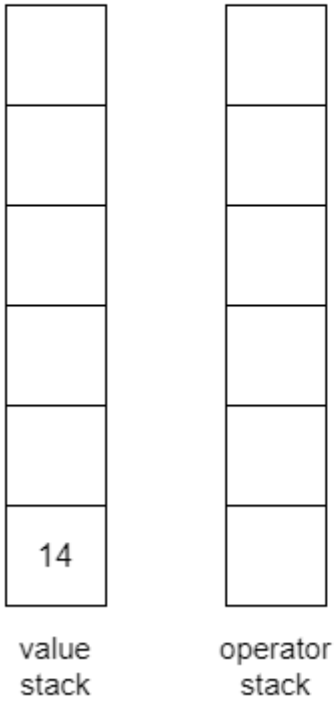
1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
      1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
      1.2.3 A right parenthesis '**)**':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
      1.2.4 A '**+**' operator or a '**-**' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
      1.2.5 A '*****' operator or a '**/**' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

1+2*6-2+(9-6)



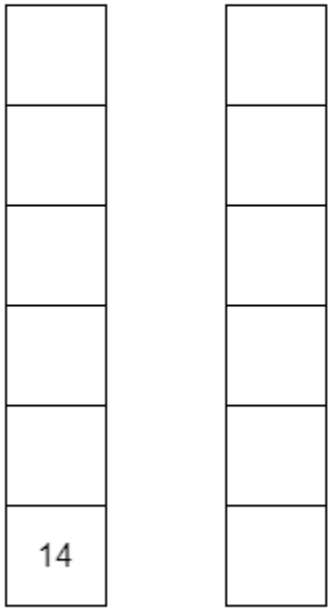
1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
      1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
      1.2.3 A right parenthesis '**)**':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
      1.2.4 A '**+**' operator or a '**-**' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
      1.2.5 A '*****' operator or a '**/**' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

1+2*6-2+(9-6)



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
      1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
      1.2.3 A right parenthesis '**)**':
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
      1.2.4 A '**+**' operator or a '**-**' operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
      1.2.5 A '*****' operator or a '**/**' operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`

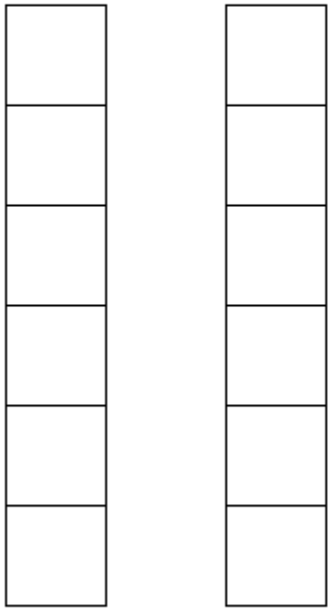
1+2*6-2+(9-6)



1. While there are still characters left in the string,
  1.1 Get the next character.
  1.2 If the character is:
    1.2.1 A digit '**0**' to '**9**': push it onto the **value stack**.
    1.2.2 A left parenthesis '**(**': push it onto the **operator stack**.
    1.2.3 A right parenthesis '**)**':
      1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis '**(**',
        1.2.3.1.1 Pop the operator from the **operator stack**.
        1.2.3.1.2 Pop the **value stack** twice, getting two operands.
        1.2.3.1.3 Apply the operator to the operands, in the correct order.
        1.2.3.1.4 Push the result onto the **value stack**.
      1.2.3.2 Pop the left parenthesis '**(**' from the **operator stack**, and discard it.
    1.2.4 A '**+**' operator or a '**-**' operator:
      1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis '**(**',
        1.2.4.1.1 Pop the operator from the **operator stack**.
        1.2.4.1.2 Pop the **value stack** twice, getting two operands.
        1.2.4.1.3 Apply the operator to the operands, in the correct order.
        1.2.4.1.4 Push the result onto the **value stack**.
      1.2.4.2 Push the '**+**' or '**-**' onto the **operator stack**.
    1.2.5 A '*****' operator or a '**/**' operator:
      1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a '*****' or a '**/**'),
        1.2.5.1.1 Pop the operator from the **operator stack**.
        1.2.5.1.2 Pop the **value stack** twice, getting two operands.
        1.2.5.1.3 Apply the operator to the operands, in the correct order.
        1.2.5.1.4 Push the result onto the **value stack**.
      1.2.5.2 Push the '*****' or '**/**' onto the **operator stack**.
2. While the **operator stack** is not empty,
  2.1 Pop the operator from the **operator stack**.
  2.2 Pop the **value stack** twice, getting two operands.
  2.3 Apply the operator to the operands, in the correct order.
  2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
  `Pop it and print the value.`

1+2*6-2+(9-6)

Ans = 14



1. While there are still characters left in the string,
   1.1 Get the next character.
   1.2 If the character is:
      1.2.1 A digit **'0'** to **'9'**: push it onto the **value stack**.
      1.2.2 A left parenthesis **'('**: push it onto the **operator stack**.
      1.2.3 A right parenthesis **')'**:
         1.2.3.1 While the thing on top of the **operator stack** is not a left parenthesis **'('**,
            1.2.3.1.1 Pop the operator from the **operator stack**.
            1.2.3.1.2 Pop the **value stack** twice, getting two operands.
            1.2.3.1.3 Apply the operator to the operands, in the correct order.
            1.2.3.1.4 Push the result onto the **value stack**.
         1.2.3.2 Pop the left parenthesis **'('** from the **operator stack**, and discard it.
      1.2.4 A **'+'** operator or a **'-'** operator:
         1.2.4.1 While the **operator stack** is not empty, and the top thing on the operator stack is not a left parenthesis **'('**,
            1.2.4.1.1 Pop the operator from the **operator stack**.
            1.2.4.1.2 Pop the **value stack** twice, getting two operands.
            1.2.4.1.3 Apply the operator to the operands, in the correct order.
            1.2.4.1.4 Push the result onto the **value stack**.
         1.2.4.2 Push the **'+'** or **'-'** onto the **operator stack**.
      1.2.5 A **'*'** operator or a **'/'** operator:
         1.2.5.1 While the **operator stack** is not empty, and (the top thing on the operator stack is a **'*'** or a **'/'**),
            1.2.5.1.1 Pop the operator from the **operator stack**.
            1.2.5.1.2 Pop the **value stack** twice, getting two operands.
            1.2.5.1.3 Apply the operator to the operands, in the correct order.
            1.2.5.1.4 Push the result onto the **value stack**.
         1.2.5.2 Push the **'*'** or **'/'** onto the **operator stack**.
2. While the **operator stack** is not empty,
   2.1 Pop the operator from the **operator stack**.
   2.2 Pop the **value stack** twice, getting two operands.
   2.3 Apply the operator to the operands, in the correct order.
   2.4 Push the result onto the **value stack**.
3. At this point the **operator stack** should be empty, and the **value stack** should have only one value in it, which is the final result.
   `Pop it and print the value.`