Applications of stack

Algorithm 1 InfixToPostfix

Require: Infix expression E
Ensure: Postfix expression P
1: Initialize empty stack S
2: Initialize empty output string P
3: **for** each token x in E **do**
4: **if** x is an operand **then**
5: Append x to P
6: **else if** x is '(' **then**
7: Push x onto S
8: **else if** x is ')' **then**
9: **while** top of S is not '(' **do**
10: Append pop(S) to P
11: **end while**
12: Pop '(' from S
13: **else if** x is an operator **then**
14: **while** S is not empty AND precedence(top(S)) $\geq$ precedence(x) **do**
15: Append pop(S) to P
16: **end while**
17: Push x onto S
18: **end if**
19: **end for**
20: **while** S is not empty **do**
21: Append pop(S) to P
22: **end while**
23: **return** P

Algorithm 2 EvaluatePostfix

Require: Postfix expression E
Ensure: Value of the expression
1: Initialize empty stack S
2: **for** each token x in E **do**
3: **if** x is an operand **then**
4: Push x onto S
5: **else if** x is an operator **then**
6: $b \leftarrow \text{pop}(S)$
7: $a \leftarrow \text{pop}(S)$
8: $result \leftarrow a\ x\ b$
9: Push $result$ onto S
10: **end if**
11: **end for**
12: **return** pop(S)