Infix to Postfix (using Stacks)

Infix expression: The expression of the form a op b. When an operator is in-between every pair of operands.

Postfix expression: The expression of the form a b op. When an operator is followed for every pair of operands.

Why postfix representation of the expression?
The compiler scans the expression either from left to right or from right to left.

Consider the below expression: a op1 b op2 c op3 d
If op1 = +, op2 = *, op3 = +

The compiler first scans the expression to evaluate the expression b * c, then again scan the expression to add a to it. The result is then added to d after another scan.

The repeated scanning makes it very in-efficient. It is better to convert the expression to postfix (or prefix) form before evaluation.

The corresponding expression in postfix form is: abc*+d+. The postfix expressions can be evaluated easily using a stack. We will cover postfix expression evaluation in a separate post.

Rough Sketch of the Algorithm (Pseudocode)
1. Scan the infix expression from left to right.
2. If the scanned character is an operand, output it.
3. Else,
…..**3.1** If the precedence of the scanned operator is greater than the precedence of the operator in the stack(or the stack is empty), push it.
…..**3.2** Else, Pop the operator from the stack until the precedence of the scanned operator is less-equal to the precedence of the operator residing on the top of the stack. Push the scanned operator to the stack.
4. If the scanned character is an ‘(‘, push it to the stack.
5. If the scanned character is an ‘)’, pop and output from the stack until an ‘(‘ is encountered.
6. Repeat steps 2-6 until infix expression is scanned.
7. Pop and output from the stack until it is not empty.