Infix to Postfix Conversion

Pseudocode

I is infix arithmetic expression
P is postfix arithmetic expression
S is a stack of operators, ***including*** "(". Parenthesis has highest preference in BODMAS.

expression Postfix(infix_expression I)

```
S.push("(")
add ")" at the end of I

while ( S.not_empty() )
      E = left-most unread entity in I

      if ( E is operand )
            P.add(E)
      else if ( E == "(" )
            S.push(E)
      else if ( E is operator )
            while ( S.top.operator_pref >= E.operator_pref)
                  // "(" has highest preference in BODMAS
                  Op=S.pop()
                  P.add(Op)
                  S.push(E)
      else if ( E == ")" )
            while ( S.top != ")" )
                  Op=S.pop()
                  P.add(Op)
            Op=S.top() // discard ")"

return P
```

Example

I: 5 * (6 + 2) – 12 / 4

Adding ")" to I results in: 5 * (6 + 2) – 12 / 4)

Entity	Stack S	Expression P
	(	
5	(,	5
*	(,*	5
(	(,*,(	5
6	(,*,(	5,6
+	(,*,(,+	5,6
2	(,*,(,+	5,6,2
)	(,*	5,6,2,+
-	(,-	5,6,2,+,*
12	(,-	5,6,2,+,*,12
/	(,-,/	5,6,2,+,*,12
4	(,-,/	5,6,2,+,*,12,4
)		5,6,2,+,*,12,4,/,-

Evaluation of Postfix expression

Pseudocode

P is postfix arithmetic expression
S is a stack of numbers

```
number post_eval(postfix_expression P)

      add ")" at the end of P

      while (P ! empty)
            E = left-most unread entity in P // (operand or
      operator)

            if ( E is operand )
                  S.push(E)
            else // this is operator
                  a = S.pop
                  b = S.pop
                  c = b E a
                  S.push(c)

      return S.top
```

Example

P: 5, 6, 2, +, *, 12, 4, /, -

Adding “)” P results in P: 5, 6, 2, +, *, 12, 4, /, -, “)”

Entity	Stack S
5	5
6	5,6
2	5,6,2
+	5,8
*	40
12	40,12
4	40,12,4
/	40,3
-	37
)	