

---

**Algorithm 1** in-fix to post-fix conversion

---

```
1: function SHUNTINGYARD(tokenList[])
2:   while tokenList.ISEMPTY() = FALSE do
3:     token  $\leftarrow$  tokenList.POP();
4:     if token.ISNUM() = TRUE then
5:       numStack.PUSH(token.GETVAL())
6:     else if token.ISVAR() = TRUE then
7:       value  $\leftarrow$  GETVAR(token)
8:       numStack.PUSH(value)
9:     else if token.ISLEFTPAREN() = TRUE then
10:      opStack.PUSH(token.GETVAL())
11:    else if token.ISRIGHTPAREN() = TRUE then
12:      while opStack.PEEK()  $\neq$  ' ( ' do
13:        op  $\leftarrow$  opStack.POP()
14:        num1  $\leftarrow$  numStack.POP()
15:        num2  $\leftarrow$  numStack.POP()
16:        result  $\leftarrow$  EVAL(op, num1, num2)
17:        numStack.PUSH(result)
18:      end while
19:      opStack.POP()
20:    else if token.ISOPERATOR() = TRUE then
21:      while opStack.ISEMPTY() = FALSE and OPPEC(opstack.PEEK(), token) = TRUE do
22:        op  $\leftarrow$  opStack.POP()
23:        num1  $\leftarrow$  numStack.POP()
24:        num2  $\leftarrow$  numStack.POP()
25:        result  $\leftarrow$  EVAL(op, num1, num2)
26:        numStack.PUSH(result)
27:      end while
28:      opStack.PUSH(token)
29:    end if
30:  end while
31:  while opStack.ISEMPTY() = FALSE do
32:    op  $\leftarrow$  opStack.POP()
33:    num1  $\leftarrow$  numStack.POP()
34:    num2  $\leftarrow$  numStack.POP()
35:    result  $\leftarrow$  EVAL(op, num1, num2)
36:    numStack.PUSH(result)
37:  end while
38:  return numStack.POP()
39: end function
```

---

---

**Algorithm 2** determine operator precedence: TRUE if *op1* has precedence

---

```
1: function OPPREC(op1, op2)
2:   if op2 = ' ( ' OR op2 = ' ) ' then
3:     return FALSE
4:   else if (op1 = ' / ' OR op1 = ' * ') AND (op2 = ' + ' OR op2 = ' - ') then
5:     return FALSE
6:   else
7:     return TRUE
8:   end if
9: end function
```

---

---

**Algorithm 3** evaluate math

---

```
1: function EVAL(op, num1, num2)
2:   if op = ' + ' then
3:     return (num1 + num2)
4:   else if op = ' - ' then
5:     return (num1 - num2)
6:   else if op = ' * ' then
7:     return (num1 * num2)
8:   else if op = ' / ' then
9:     if num2 = 0 then
10:      return ERROR
11:    else
12:      return (num1 / num2)
13:    end if
14:   end if
15: end function
```

---