

```

1  #include <iostream>
2  using namespace std ;
3
4  struct Node
5  {
6      int      operand    ;
7      char     operation  ;
8      struct Node * left  ;
9      struct Node * right ;
10 };
11
12 typedef struct Node * Tree ;
13
14 void postfix_display(Tree tree)
15 {
16     if(tree != NULL)
17     {
18         postfix_display(tree->left)    ;
19         postfix_display(tree->right)   ;
20         if(tree->operation == '\0')
21             cout << " " << tree->operand    ;
22         else
23             cout << " " << tree->operation  ;
24     }
25
26     return ;
27 }
28
29 void prefix_display(Tree tree)
30 {
31     if(tree != NULL)
32     {
33         if(tree->operation == '\0')
34             cout << " " << tree->operand    ;
35         else
36             cout << " " << tree->operation  ;
37         prefix_display(tree->left)    ;
38         prefix_display(tree->right)   ;
39     }
40
41     return ;
42 }
43
44 void expression_infix_display(Tree tree)
45 {
46     if(tree != NULL)
47     {
48         if((tree->left == NULL)&&(tree->right == NULL))
49         {
50             expression_infix_display(tree->left)    ;
51             cout << tree->operand    ;
52             expression_infix_display(tree->right)   ;
53         }
54         else
55         {
56             cout << '('    ;
57             expression_infix_display(tree->left)    ;
58             cout << tree->operation  ;
59             expression_infix_display(tree->right)   ;
60             cout << ')'    ;
61         }
62     }
63
64     return ;
65 }
66
67 int expression_evaluation(Tree tree)
68 {
69     if((tree->left == NULL)&&(tree->right == NULL))
70         return tree->operand ;
71     else
72     {
73         switch(tree->operation)
74         {
75             case '+' :
76                 return expression_evaluation(tree->left) + expression_evaluation(tree->right) ;
77         }
78     }
79 }

```

```

76         case '-' :
76         return expression_evaluation(tree->left) - expression_evaluation(tree->right) ;
77         case '*' :
77         return expression_evaluation(tree->left) * expression_evaluation(tree->right) ;
78         case '/' :
78         return expression_evaluation(tree->left) / expression_evaluation(tree->right) ;
79     }
80 }
81 }
82
83 int main()
84 {
85     Tree n0 = new Node ;
86
87     n0->operation = '*' ;
88     n0->operand = 0 ;
89
90     Tree n11 = new Node ;
91
92     n11->operation = '\0' ;
93     n11->operand = 3 ;
94
95     Tree n12 = new Node ;
96
97     n12->operation = '/' ;
98     n12->operand = 0 ;
99
100    Tree n21 = new Node ;
101
102    n21->operation = '+' ;
103    n21->operand = 0 ;
104
105    Tree n22 = new Node ;
106
107    n22->operation = '\0' ;
108    n22->operand = 2 ;
109
110    Tree n31 = new Node ;
111
112    n31->operation = '\0' ;
113    n31->operand = 5 ;
114
115    Tree n32 = new Node ;
116
117    n32->operation = '\0' ;
118    n32->operand = 1 ;
119
120    n0->left = n11 ;
121    n0->right = n12 ;
122
123    n11->left = NULL ;
124    n11->right = NULL ;
125
126    n12->left = n21 ;
127    n12->right = n22 ;
128
129    n21->left = n31 ;
130    n21->right = n32 ;
131
132    n22->left = NULL ;
133    n22->right = NULL ;
134
135    n31->left = NULL ;
136    n31->right = NULL ;
137
138    n32->left = NULL ;
139    n32->right = NULL ;
140
141    postfix_display(n0) ; cout << endl ;
142    prefix_display(n0) ; cout << endl ;
143
144    cout << "3*((5+1)/2) = " << expression_evaluation(n0) << endl ;
145
146    expression_infix_display(n0) ;
147
148    return 0 ;
149 }

```