```cpp
#include <iostream>
using  namespace  std  ;

struct  Node
{
    int           operand    ;
    char          operation  ;
    struct Node * left   ;
    struct Node * right  ;
};

typedef  struct Node *  Tree  ;

void  postfix_display(Tree tree)
{
    if(tree != NULL)
    {
        postfix_display(tree->left)    ;
        postfix_display(tree->right)   ;
        if(tree->operation == '\0')
            cout  <<  " "  <<  tree->operand    ;
        else
            cout  <<  " "  <<  tree->operation  ;
    }

    return  ;
}

void  prefix_display(Tree tree)
{
    if(tree != NULL)
    {
        if(tree->operation == '\0')
            cout  <<  " "  <<  tree->operand    ;
        else
            cout  <<  " "  <<  tree->operation  ;
        prefix_display(tree->left)     ;
        prefix_display(tree->right)    ;
    }

    return  ;
}

void  expression_infix_display(Tree tree)
{
    if(tree != NULL)
    {
        if((tree->left == NULL)&&(tree->right == NULL))
        {
            expression_infix_display(tree->left)   ;
            cout  <<  tree->operand                ;
            expression_infix_display(tree->right)  ;
        }
        else
        {
            cout  <<  '('                          ;
            expression_infix_display(tree->left)   ;
            cout  <<  tree->operation              ;
            expression_infix_display(tree->right)  ;
            cout  <<  ')'                          ;
        }
    }

    return  ;
}

int  expression_evaluation(Tree tree)
{
    if((tree->left == NULL)&&(tree->right == NULL))
        return  tree->operand  ;
    else
    {
        switch(tree->operation)
        {
            case '+' :
            return  expression_evaluation(tree->left) + expression_evaluation(tree->right)  ;

            case '-' :
            return  expression_evaluation(tree->left) - expression_evaluation(tree->right)  ;
            case '*' :
            return  expression_evaluation(tree->left) * expression_evaluation(tree->right)  ;
            case '/' :
            return  expression_evaluation(tree->left) / expression_evaluation(tree->right)  ;
        }
    }
}

int  main()
{
    Tree  n0  =  new  Node  ;

    n0->operation  =  '*'  ;
    n0->operand    =  0    ;

    Tree  n11  =  new  Node  ;

    n11->operation  =  '\0'  ;
    n11->operand    =  3     ;

    Tree  n12  =  new  Node  ;

    n12->operation  =  '/'  ;
    n12->operand    =  0    ;

    Tree  n21  =  new  Node  ;

    n21->operation  =  '+'  ;
    n21->operand    =  0    ;

    Tree  n22  =  new  Node  ;

    n22->operation  =  '\0'  ;
    n22->operand    =  2     ;

    Tree  n31  =  new  Node  ;

    n31->operation  =  '\0'  ;
    n31->operand    =  5     ;

    Tree  n32  =  new  Node  ;

    n32->operation  =  '\0'  ;
    n32->operand    =  1     ;

    n0->left   =  n11  ;
    n0->right  =  n12  ;

    n11->left   =  NULL  ;
    n11->right  =  NULL  ;

    n12->left   =  n21  ;
    n12->right  =  n22  ;

    n21->left   =  n31  ;
    n21->right  =  n32  ;

    n22->left   =  NULL  ;
    n22->right  =  NULL  ;

    n31->left   =  NULL  ;
    n31->right  =  NULL  ;

    n32->left   =  NULL  ;
    n32->right  =  NULL  ;

    postfix_display(n0)  ;  cout  <<  endl  ;
    prefix_display(n0)   ;  cout  <<  endl  ;

    cout  <<  "3*((5+1)/2) = "   <<  expression_evaluation(n0)   <<  endl  ;

    expression_infix_display(n0)  ;

    return  0  ;
}
```