```
#include <iostream>
#include <string.h>
using namespace std;

struct node
{
    char data;
    node *left;
    node *right;
};
class tree
{
    char prefix[20];

public:
    node *top;
    void expression(char[]);
    void display(node *);
    void non_rec_postorder(node *);
    void del(node *);
};
class stack1
{
    node *data[30];
    int top;

public:
    stack1()
    {
        top = -1;
    }
    int empty()
    {
        if (top == -1)
            return 1;
        return 0;
    }
    void push(node *p)
    {
        data[++top] = p;
    }
    node *pop()
    {
        return (data[top--]);
    }
};
void tree::expression(char prefix[])
{
    char c;
    stack1 s;
    node *t1, *t2;
    int len, i;
    len = strlen(prefix);
    for (i = len - 1; i >= 0; i--)
    {
        top = new node;
        top->left = NULL;
        top->right = NULL;
        if (isalpha(prefix[i]))
        {
            top->data = prefix[i];
            s.push(top);
        }
        else if (prefix[i] == '+' || prefix[i] == '*' || prefix[i] == '-' ||
prefix[i] == '/')
        {
            t2 = s.pop();
            t1 = s.pop();
            top->data = prefix[i];
            top->left = t2;
            top->right = t1;
            s.push(top);
        }
    }
    top = s.pop();
}
void tree::display(node *root)
{
    if (root != NULL)
    {
        cout << root->data;
        display(root->left);
        display(root->right);
    }
}
void tree::non_rec_postorder(node *top)
{
    stack1 s1, s2;
    node *T = top;
    cout << "\n";
    s1.push(T);
    while (!s1.empty())
    {
        T = s1.pop();
        s2.push(T);
        if (T->left != NULL)
            s1.push(T->left);
        if (T->right != NULL)
            s1.push(T->right);
    }
    while (!s2.empty())
    {
        top = s2.pop();
        cout << top->data;
    }
}
void tree::del(node *node)
{
    if (node == NULL)
        return;

    del(node->left);
    del(node->right);

    cout <<endl<<"Deleting node : " << node->data<<endl;

}
```

```
int main()
{
    char expr[20];
    tree t;

    cout <<"Enter prefix Expression : ";
    cin >> expr;
    cout << expr;
    t.expression(expr);

    t.non_rec_postorder(t.top);
    t.del(t.top);

}
```