```cpp
#include <iostream>
#include <cstring>  // Use <cstring> for string functions in C++98
using namespace std;

struct node
{
    char data;
    node *left;
    node *right;
};

class tree
{
    char prefix[20];

public:
    node *root; // Renamed top to root to avoid conflict with stack

    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;  // Initialize stack top to -1.
    }

    int empty()
        {
        return top == -1;
    }

    void push(node *p)
        {
        if (top < 29)
                {
            data[++top] = p;
        }
    }

    node* pop()
        {
        if (top >= 0)
                {
            return data[top--];
        }
```

```
        return NULL; // Use NULL instead of nullptr in C++98
    }
};

void tree::expression(char prefix[])
{
    char c;
    stack1 s;
    node *t1, *t2;
    int len = strlen(prefix);

    for (int i = len - 1; i >= 0; i--)
        {
        node *top = new node; // Local variable, so no conflict with class member root
        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] == '/')
                {
            t1 = s.pop();
            t2 = s.pop();
            top->data = prefix[i];
            top->left = t1;
            top->right = t2;
            s.push(top);
        }
    }

    root = s.pop();  // Set the root of the expression tree to the last node in the stack.
}

void tree::display(node *root) {
    if (root == NULL) return;

    display(root->left);
    cout << root->data << " ";
    display(root->right);
}

void tree::non_rec_postorder(node *root)
{
    if (root == NULL) return;

    stack1 s1, s2;
    s1.push(root);

    while (!s1.empty())
        {
        node *t = s1.pop();
```

```
110         s2.push(t);
111
112         if (t->left != NULL)
113                         s1.push(t->left);
114         if (t->right != NULL)
115                         s1.push(t->right);
116     }
117
118     while (!s2.empty())
119         {
120         node *t = s2.pop();
121         cout << t->data << " ";
122     }
123 }
124
125 void tree::del(node *root)
126 {
127     if (root == NULL) return;
128
129     del(root->left);
130     del(root->right);
131
132     cout << "Deleting node: " << root->data << endl;
133     delete root;  // Delete the node.
134 }
135
136 int main()
137 {
138     char expr[20];
139     tree t;
140
141     cout << "Enter prefix Expression: ";
142     cin >> expr;
143
144     t.expression(expr);
145
146     cout << "Postorder Traversal: ";
147     t.non_rec_postorder(t.root); // Use root instead of top
148     cout << endl;
149
150     t.del(t.root); // Use root instead of top
151     return 0;
152 }
153
```

OutPut :


D:\SE Computer\LAB CODES\DSA\DSA4.exe
Enter prefix Expression: +--a*bc/def
Prefix Expression is : +--a*bc/def
Postorder Traversal: a b c * - d e / - f +
Deleting node: a
Deleting node: b
Deleting node: c
Deleting node: *
Deleting node: -
Deleting node: d
Deleting node: e
Deleting node: /
Deleting node: -
Deleting node: f
Deleting node: +

Process exited after 20.83 seconds with return value 0
Press any key to continue . . .