```cpp
#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 stack
{
    node *data[30];
    int top;
public:
    stack()
    {
        top = -1;
    }
    int isempty()
    {
        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;
    stack s;
    node *t1, *t2;
    int len, i;
    len = strlen(prefix);
    for (i = len - 1; i >= 0; i--)
    {
        top = new node;
```

```
54          top->left = NULL;
55          top->right = NULL;
56          if (isalpha(prefix[i]))
57          {
58              top->data = prefix[i];
59              s.push(top);
60          }
61          else if (prefix[i] == '+' || prefix[i] == '*' || prefix[i] == '-' ||
    prefix[i] == '/')
62          {
63              t2 = s.pop();
64              t1 = s.pop();
65              top->data = prefix[i];
66              top->left = t2;
67              top->right = t1;
68              s.push(top);
69          }
70      }
71      top = s.pop();
72  }
73  void tree::display(node *root)
74  {
75      if (root != NULL)
76      {
77          cout<< root->data;
78          display(root->left);
79          display(root->right);
80      }
81  }
82  void tree::non_rec_postorder(node *top)
83  {
84      stack s1, s2;
85      node *T = top;
86      s1.push(T);
87      while (!s1.isempty())
88      {
89          T = s1.pop();
90          s2.push(T);
91          if (T->left != NULL)
92              s1.push(T->left);
93          if (T->right != NULL)
94              s1.push(T->right);
95      }
96      while (!s2.isempty())
97      {
98          top = s2.pop();
99          cout <<" | "<<top->data;
100     }
101 }
102 void tree::del(node *node)
103 {
104     if (node == NULL)
105         return;
106     del(node->left);
107     del(node->right);
```

```
108     cout<<endl<<"Deleting Node "<< node→data<<endl;
109 }
110 int main()
111 {
112     char expr[20];
113     tree t;
114     cout <<"Enter Prefix Expression ";
115     cin>>expr;
116     cout<<endl<<"Stack"<<endl;
117     t.expression(expr);
118     cout<<endl;
119     t.non_rec_postorder(t.top);
120     cout<<endl;
121     t.del(t.top);
122     cout<<endl<<"Original Expression ";
123     t.display(t.top);
124     cout<<endl;
125 }
126
```