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

class Node
{
   public:
      char data;
      Node* left;
      Node* right;
};

class stack
{
   Node* stack1[30];
   int top;
   public:
   stack()
   {
      top=-1;
   }
   bool is_empty()
   {
      if(top==-1)
         return 1;
      else
         return 0;
   }

   void push(Node* t)
   {
        ++top;
        stack1[top]=t;
    }

    Node* pop()
    {

        if(!is_empty())
        {
            Node* t=stack1[top];
            top--;
            return t;
        }
        else
            return NULL;
    }

};

class Tree
{
    //char prefix[30];
    public:
    Node* root;
        void expression(char exp[])
        {
            Node* t1;
            Node* t2;
            stack s;
            int len=strlen(exp);
            for(int i=len;i>=0;i--)
    {
        root=new Node;
        root->left=root->right=NULL;
        if(isalpha(exp[i]))
        {
            root->data=exp[i];
            s.push(root);
        }

        else if(exp[i]=='+' || exp[i]=='-' || exp[i]=='/' || exp[i]=='*')
        {
            t1=s.pop();
            t2=s.pop();
            root->data=exp[i];
            root->left=t1;
            root->right=t2;
            s.push(root);
        }
    }
    root=s.pop();
}

void postorder(Node* root)
{
    stack s1,s2;
    Node* t=root;
    s1.push(t);
    while(!s1.is_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.is_empty())
            {
                t=s2.pop();
                cout<<t->data;
            }
            cout<<"\n";
        }

        void del(Node* root)
        {
            if(root==NULL)
                return;
            del(root->left);
            del(root->right);
            cout<<"Deleted node-> "<<root->data<<endl;
            free(root);
        }
};

int main()
{
    char express[20];
    Tree t1;
    int c;
    while(1)
```

```
 {
    cout<<"1-Entering the expression.\n2-Printing normally.\n3-Print using non recursive
postorder.\n4-Deleting the tree.\n5-exit"<<endl;
    cout<<"Enter your choice:"<<endl;
    cin>>c;
    if(c==1)
    {
       cout<<"Enter the prefix expression:"<<endl;
       cin>>express;
       t1.expression(express);
    }
    else if(c==2)
    {
       cout<<express<<endl;
    }
    else if(c==3)
    {
       t1.postorder(t1.root);
    }
    else if(c==4)
    {
       t1.del(t1.root);
    }
    else if(c==5)
    {
       cout<<"End of program."<<endl;
       break;
    }
    else
       cout<<"Wrong choice!!!"<<endl;
 }
```

```
}
```

OUTPUT:-

```
1-Entering the expression.
2-Printing normally.
3-Print using non recursive postorder.
4-Deleting the tree.
5-exit
Enter your choice:
1
Enter the prefix expression:
+a*bc
1-Entering the expression.
2-Printing normally.
3-Print using non recursive postorder.
4-Deleting the tree.
5-exit
Enter your choice:
2
+a*bc
1-Entering the expression.
2-Printing normally.
3-Print using non recursive postorder.
4-Deleting the tree.
5-exit
Enter your choice:
3
abc*+
1-Entering the expression.
2-Printing normally.
```

```
3-Print using non recursive postorder.

4-Deleting the tree.

5-exit

Enter your choice:

4

Deleted node-> a

Deleted node-> b

Deleted node-> c

Deleted node-> *

Deleted node-> +

1-Entering the expression.

2-Printing normally.

3-Print using non recursive postorder.

4-Deleting the tree.

5-exit

Enter your choice:

5

End of program.
```