Assignment no.5

INPUT:
```
#include <iostream>
#include <stack>
#include<string>
using namespace std;

class BTNODE
{
     BTNODE * left,*right;
     char data;
  public:
       BTNODE()
      {
             left=right=NULL;
             data='\0';
      }
     friend class ExpressionTree;
};

class ExpressionTree
{
    BTNODE * root;
  public:
    ExpressionTree()
    {
        root=NULL;
    }
    void create(string);
    void Inorder(BTNODE *);
    void Preorder(BTNODE *);
    void Postorder(BTNODE *);
    void Delete_tree(BTNODE *);
    friend int main();
};

void ExpressionTree::create(string s)
{
    stack<BTNODE *> s1;
    BTNODE *T1,*T2,*T3;
    int i;
    for(i=s.length()-1;i>=0;i--)
    {
        if(isalnum(s[i]))
        {
            T1=new BTNODE;
            T1->data=s[i];
            s1.push(T1);
        }
        else
        {
           T3=s1.top();
           s1.pop();
           T2=s1.top();
           s1.pop();
           T1=new BTNODE;
           T1->data=s[i];
           T1->left=T3;
           T1->right=T2;
           s1.push(T1);
        }
    }
    root=s1.top();
    s1.pop();
}

void ExpressionTree::Inorder(BTNODE *T)
{
    if(T!=NULL)
    {
     Inorder(T->left);
     cout<<"\t "<<T->data;
     Inorder(T->right);
    }
}

void ExpressionTree::Preorder(BTNODE *T)
{
    if(T!=NULL)
    {
     cout<<"\t "<<T->data;
     Preorder(T->left);
     Preorder(T->right);
    }
}

void ExpressionTree::Postorder(BTNODE *T)
{
     stack<BTNODE*> s2;
     BTNODE * prev=NULL;
     do
     {
        while(T!=NULL)
        {
           s2.push(T);
           T=T->left;
        }
        T=s2.top();
        if(T->right==NULL ||T->right==prev)
        {
           cout<<"\t "<<T->data;
           prev=s2.top();
           s2.pop();
           T=NULL;
        }
        else
           T=T->right;
     }while(!s2.empty());
}

void ExpressionTree::Delete_tree(BTNODE *T)
{
    if(T!=NULL)
    {
     Delete_tree(T->left);
     Delete_tree(T->right);
     delete T;
    }
}

int main()
```

```
{
    ExpressionTree E;
      string s;
      int ch;
      do
      {
           cout<<"\n ---------------Menu-------------";
           cout<<"\n 1.Create Expression Tree";
           cout<<"\n 2.Infix Expression";
           cout<<"\n 3.Prefix Expression";
           cout<<"\n 4.Postfix Expression";
           cout<<"\n 5.Delete a Tree";
           cout<<"\n 6.Exit";
           cout<<"\n --------------------------------";
           cout<<"\n Enter your choice =";
           cin>>ch;
           switch(ch)
           {
               case 1:
                 cout<<"\n Enter the prefix expression=>";
                 cin>>s;
                 E.create(s);
                            break;
               case 2:
                 if(E.root==NULL)
                      cout<<"\n Tree is empty......";
                 else
                 {
                      cout<<"\n Infix Expression=>";
                     E.Inorder(E.root);
                 }
                           break;
               case 3:
                 if(E.root==NULL)
                      cout<<"\n Tree is empty......";
                 else
                 {
                      cout<<"\n Prefix Expression=>";
                     E.Preorder(E.root);
                 }
                           break;
               case 4:
                 if(E.root==NULL)
                      cout<<"\n Tree is empty......";
                 else
                 {
                      cout<<"\n Postfix Expression=>";
                      E.Postorder(E.root);
                 }
                           break;
               case 5:
                if(E.root==NULL)
                      cout<<"\n Tree is empty......";
                    else
                    {
                      E.Delete_tree(E.root);
                      cout<<"\n Tree deleted.......";
                      E.root=NULL;
                    }
                           break;
               case 6:
```

```
                              break;
                 default:
                   cout<<"\n Enter Correct Choice.......!";
            }
      }while(ch != 6);

      return 0;
}
```
OUTPUT:

```
---------------Menu-------------

 1.Create Expression Tree

 2.Infix Expression

 3.Prefix Expression

 4.Postfix Expression

 5.Delete a Tree

6.Exit --------------------------------

 Enter your choice =1

 Enter the prefix expression=>+-abc

 ---------------Menu-------------

 1.Create Expression Tree

 2.Infix Expression

 3.Prefix Expression

 4.Postfix Expression

 5.Delete a Tree

 6.Exit--------------------------------

 Enter your choice =2

 Infix Expression=>      a      -      b      +      c

 ---------------Menu-------------

 1.Create Expression Tree

 2.Infix Expression

 3.Prefix Expression

 4.Postfix Expression

 5.Delete a Tree

 6.Exit
```

```
--------------------------------
Enter your choice =3
Prefix Expression=>     +       -       a       b       c
---------------Menu-------------
1.Create Expression Tree
2.Infix Expression
3.Prefix Expression
4.Postfix Expression
5.Delete a Tree
6.Exit
--------------------------------
Enter your choice =4
Postfix Expression=>    a       b       -       c       +
---------------Menu-------------
1.Create Expression Tree
2.Infix Expression
3.Prefix Expression
4.Postfix Expression
5.Delete a Tree
6.Exit
--------------------------------
Enter your choice =5
Tree deleted.......---------------Menu-------------
1.Create Expression Tree
2.Infix Expression
3.Prefix Expression
4.Postfix Expression
5.Delete a Tree
6.Exit-------------------------------
Enter your choice =6
```