Assignment 5

Problem Statement:-

Construct an expression tree from the given prefix expression eg. +--a*bc/def and traverse it using post order traversal (non recursive) and then delete the entire tree.

Source Code:-

```
#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;
   }

   bool isempty()
   {
      return top == -1;
   }
```

```
    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;
        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)
{
   stack s1, s2;
   node* T = top;
   s1.push(T);

   while (!s1.isempty())
   {
      T = s1.pop();
      s2.push(T);

      if (T->left != NULL) s1.push(T->left);
      if (T->right != NULL) s1.push(T->right);
   }

   while (!s2.isempty())
   {
      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 << endl << "Building Tree..." << endl;

    t.expression(expr);

    cout << endl << "Postorder Traversal: ";
    t.non_rec_postorder(t.top);
    cout << endl;

    cout << "Deleting Nodes in Postorder:" << endl;
    t.del(t.top);

    cout << endl << "Original Expression: ";
    t.display(t.top);
    cout << endl;

    return 0;
}
```

Output:-