**Assignment No:  7**

**Problem Statement:**

Construct and expression tree from postfix/prefix expression and perform recursive and non-recursive In-order, pre-order and post-order traversals.

**Outcomes:**

Students will be able to:

1. Make use of coding standards for application development.

**Code:**

#include<iostream>

#include<ctype.h>

using namespace std;

class TreeNode{

public:

char data;

TreeNode \*left;

TreeNode \*right;

TreeNode(char ch){

data = ch;

left = NULL;

right = NULL;

}

};

class Node{

public:

TreeNode \*data;

Node \*prev;

};

class Stack{

private:

Node \*top;

public:

Stack(){

top = NULL;

}

void push(TreeNode \*treeNode){

Node \*newNode = new Node;

newNode->data = treeNode;

newNode->prev = top;

top = newNode;

}

TreeNode\* pop(){

if(top == NULL){

cout<<"\nSTACK UNDERFLOW !";

return '\0';

}

Node \*temp = top;

TreeNode\* data = temp->data;

top = top->prev;

delete temp;

return data;

}

TreeNode\* peek(){

return top->data;

}

~Stack(){

Node \*temp;

temp = top;

while(temp!=NULL){

delete temp;

temp = temp->prev;

}

delete top;

}

};

void inorderTraversal(TreeNode\* root){

if(root){

inorderTraversal(root->left);

cout<<root->data<<"\t";

inorderTraversal(root->right);

}

}

int main(){

string postfix;

Stack stk;

cout<<"Enter the postfix expression:"<<endl;

cin>>postfix;

for(int i =0;i<postfix.length();i++){

TreeNode \*newNode = new TreeNode(postfix[i]);

if(isalpha(postfix[i])){

stk.push(newNode);

}

else{

newNode->right = stk.pop();

newNode->left = stk.pop();

stk.push(newNode);

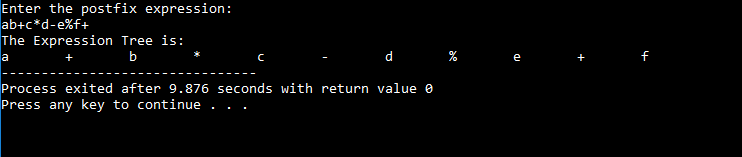
}

}

cout<<"The Expression Tree is:"<<endl;

inorderTraversal(stk.pop());

return 0;

}

**Output:**