**Assignment number: 4**

**Subject: ADVANCED DATA STRUCTURES LAB**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**PROBLEM STATEMENT:**

For given expression eg. a-b\*c-d/e+f construct inorder sequence and traverse it using postorder traversal(non recursive).

**Code:**

#include<iostream>

#include<string.h>

#include<stack>

using namespace std;

class node

{

char data;

node \*left,\*right;

friend class infix;

};

node \*t1,\*t2,\*t3,\*t,\*p;

class infix

{

public:

stack<node \*>st;

int weight;

int i=0,k=0;

char postfix[100];

char ch;

int size;

void infix\_to\_postfix(char infix[],int size)

{

stack<char>s;

while(i<size)

{

ch=infix[i];

if(ch=='(')

{

s.push(ch);

i++;

continue;

}

if(ch==')')

{

while(!s.empty() && s.top()!='(')

{

postfix[k++]=s.top();

s.pop();

}

if(s.empty())

{

s.pop();

}

i++;

continue;

}

weight=getweight(ch);

if(weight==0)

{

postfix[k++]=ch;

}

else

{

if(s.empty())

s.push(ch);

else

{

while(!s.empty() && s.top()!='(' && weight<=getweight(s.top()))

{

postfix[k++]=s.top();

s.pop();

}

s.push(ch);

}

}

i++;

}

while(!s.empty())

{

postfix[k++]=s.top();

s.pop();

}

postfix[k]=0;

cout<<"\nthe postfix expression is\n";

for(i=0;i<size;i++)

{

cout<<postfix[i];

}

cout<<endl;

}

int getweight(char ch)

{

switch(ch)

{

case '^':return 3;break;

case '\*':

case '/':

return 2;break;

case '+':

case '-':

return 1;break;

default:return 0;

}

}

int create();

int recursive(node\*);

int non\_recursive();

};

int infix::create()

{

int i;

for(i=0;postfix[i]!='\0';i++)

{

if(postfix[i]== '+' || postfix[i]== '-' || postfix[i]== '\*' || postfix[i]== '/')

{

t=new node;

t->data=postfix[i];

t->left=t->right=NULL;

t1=st.top();

st.pop();

t2=st.top();

st.pop();

t->left=t2;

t->right=t1;

st.push(t);

}

else

{

t3=new node;

t3->data=postfix[i];

t3->left=t3->right=NULL;

st.push(t3);

}

}

p=st.top();

st.pop();

}

int infix::recursive(node \*t)

{

if(t!=NULL)

{

recursive(t->left);

recursive(t->right);

cout<<t->data;

}

}

int infix::non\_recursive()

{

t=p;

if(t==NULL)

cout<<"\nNO NODE PRESENT\n";

stack<node \*> stk;

stack<bool> v;

while(t!=NULL)

{

stk.push(t);

v.push(false);

t=t->left;

}

while(!stk.empty())

{

t=stk.top();

bool enc=v.top();

if(enc==true)

{

cout<<t->data;

stk.pop();

v.pop();

}

else

{

v.pop();

v.push(true);

t=t->right;

while(t!=NULL)

{

stk.push(t);

v.push(false);

t=t->left;

}

}

}

}

int main()

{

infix obj;

char inf[100];

int size,choice;

char ans;

cout<<"\nenter the infix expression\n";

cin>>inf;

size=strlen(inf);

obj.infix\_to\_postfix(inf,size);

obj.create();

do

{

cout<<"\n1.RECURSIVE POST ORDER TRAVERSAL\n2.NON-RECURSIVE POST ORDER TRAVERSAL\n";

cin>>choice;

if(choice==1)

obj.recursive(p);

else if(choice==2)

obj.non\_recursive();

else

cout<<"\ninvalid choice...try again\n";

cout<<"\ndo you want to continue?(y/n)\n";

cin>>ans;

}while(ans=='y'|| ans=='Y');

return 0;

}

**OUTPUT:**

enter the infix expression

a+b-c\*d-f/h\*g\*w

the postfix expression is

ab+cd\*-fh/g\*w\*-

1.RECURSIVE POST ORDER TRAVERSAL

2.NON-RECURSIVE POST ORDER TRAVERSAL

1

ab+cd\*-fh/g\*w\*-

do you want to continue?(y/n)

y

1.RECURSIVE POST ORDER TRAVERSAL

2.NON-RECURSIVE POST ORDER TRAVERSAL

2

ab+cd\*-fh/g\*w\*-

do you want to continue?(y/n)

n