**Name : Resham Landge**

**Roll No : 2334**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***Assignment No : 4** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

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

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

#include<iostream>

using namespace std;

struct tree

{

char data;

struct tree \*left, \*right;

tree(char c)

{

Data = c;

left = right = NULL;

}

};

class stack

{

tree \*stk[30];

int top;

public:

stack()

{

top = -1;

}

int empty();

int full();

void push(tree\*);

tree\* pop();

};

int stack :: empty()

{

if(top == -1)

return 1;

return 0;

}

int stack :: full()

{

if(top == 29)

return 1;

return 0;

}

void stack :: push(tree \*d)

{

if( !full() )

stk[++top] = d;

}

tree\* stack :: pop()

{

tree \*temp = stk[top];

top--;

return temp;

}

tree \*create()

{

char str[100];

int i=0;

stack s;

cout<<" \n\t Enter the postfix expression : ";

cin>>str;

while(str[i]!='\0')

{

tree \*node = new tree(str[i]);

if( isalnum(str[i]) )

s.push(node);

else

{

node->right = s.pop();

node->left = s.pop();

s.push(node);

}

i++;

}

return s.pop();

}

void inorder(tree \*root)

{

if(root!=NULL)

{

inorder(root->left);

cout<<root->data;

inorder(root->right);

}

}

void preorder(tree \*root)

{

if(root!=NULL)

{

cout<<root->data;

preorder(root->left);

preorder(root->right);

}

}

void postorder(tree \*root)

{

if(root!=NULL)

{

postorder(root->left);

postorder(root->right);

cout<<root->data;

}

}

void nonpre(tree \*root)

{

stack s;

while( !s.empty() || root!=NULL )

{

while(root!=NULL)

{

cout<<root->data;

s.push(root);

root = root->left;

}

root = s.pop();

root = root->right;

}

}

void nonrin(tree \*root)

{

stack s;

while( !s.empty() || root!=NULL )

{

while(root!=NULL)

{

s.push(root);

root = root->left;

}

root = s.pop();

cout<<root->data;

root = root->right;

}

}

void nonpost(tree \*root)

{

stack s;

int i=0;

char str[100];

while( !s.empty() || root! = NULL)

{

while(root != NULL)

{

str[i++] = root->data;

s.push(root);

root = root->right;

}

root = s.pop();

root = root->left;

}

while(--i>=0)

cout<<str[i];

}

int main()

{

struct tree \*root = NULL;

char ch='y';

int choice;

root = create();

do

{

cout<<" \n\t 1. Preorder Traversal \n\t 2. Inorder Traversal \n\t 3. Postorder

Traversal \n\n\t Enter u r Choice : ";

cin>>choice;

cout<<"\n\n";

switch(choice)

{

case 1 : cout<<"\n\t -------------------------------------------- ";

cout<<"\n\t Preorder Recursive : ";

preorder(root);

cout<<"\n\t -------------------------------------------- ";

cout<<"\n\t Peorder Non-Recursive : ";

nonpre(root);

break;

case 2 : cout<<"\n\t -------------------------------------------- ";

cout<<"\n\t Inorder Recursive : ";

inorder(root);

cout<<"\n\t -------------------------------------------- ";

cout<<"\n\t Inorder Non-Recurcive : ";

nonrin(root);

break;

case 3 : cout<<"\n\t -------------------------------------------- ";

cout<<"\n\t Postorder Recursive : ";

postorder(root);

cout<<"\n\t -------------------------------------------- ";

cout<<"\n\t Postorder Non-Recursive : ";

nonpost(root);

break;

}

cout<<"\n\n\t \*\*\* Do u want to continue(y/n) : ";

cin>>ch;

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

return 1;

}

***OUTPUT :***

resham@ubuntu:~/Desktop/resham$ g++ assgn4.cpp

resham@ubuntu:~/Desktop/resham$ ./a.out

Enter the postfix expreesion : ab-cd/\*e+

1. Preorder Traversal

2. Inorder Traversal

3. Postorder Traversal

Enter u r Choice : 1

--------------------------------------------

Preorder Recursive : +\*-ab/cde

--------------------------------------------

Peorder Non-Recursive : +\*-ab/cde

\*\*\* Do u want to continue(y/n) : y

1. Preorder Traversal

2. Inorder Traversal

3. Postorder Traversal

Enter u r Choice : 2

--------------------------------------------

Inorder Recursive : a-b\*c/d+e

--------------------------------------------

Inorder Non-Recurcive : a-b\*c/d+e

\*\*\* Do u want to continue(y/n) : y

1. Preorder Traversal

2. Inorder Traversal

3. Postorder Traversal

Enter u r Choice : 3

--------------------------------------------

Postorder Recursive : ab-cd/\*e+

--------------------------------------------

Postorder Non-Recursive : ab-cd/\*e+

\*\*\* Do u want to continue(y/n) : n

resham@ubuntu:~/Desktop/resham