//dsa 4 expression tree traversal

#include<iostream>

#include<string.h>

using namespace std;

class ExpTree

{

class Node

{

public:

char Data;

Node \*Left,\*Right;

Node(char C=0){Data=C;Right=Left=NULL;}

}\*Root;

public:

ExpTree(){ Root=NULL;}

void Post\_Create(char[]);

void Pre\_Create(char[]);

void preOrder(Node\*);

void inOrder(Node\*);

void postOrder(Node\*);

void NonRpre();

void NonRin();

void NonRpost();

Node\* return\_Root(){ return Root;}

};

void ExpTree::Post\_Create(char Estr[25])

{

int I=0,Top=-1;

Node \*S[25];

while(Estr[I]!='\0')

{

Node \*nNode=new Node(Estr[I]);

if(isalnum(Estr[I])) //Operand

S[++Top]=nNode;

else //OPerator

{

nNode->Right=S[Top--];

nNode->Left=S[Top--];

S[++Top]=nNode;

}

I++;

}

Root= S[Top];

}

void ExpTree::Pre\_Create(char Estr[25])

{

int Top=-1,I=strlen(Estr);

Node \*S[25];

I--;

while(I>-1)

{

Node \*nNode=new Node(Estr[I]);

if(isalnum(Estr[I])) //Operand

S[++Top]=nNode;

else //OPerator

{

nNode->Left=S[Top--];

nNode->Right=S[Top--];

S[++Top]=nNode;

}

I--;

}

Root= S[Top];

}

void ExpTree::preOrder(Node \*Temp)

{

if(Temp!=NULL)

{

cout<<Temp->Data;

preOrder(Temp->Left);

preOrder(Temp->Right);

}

}

void ExpTree::inOrder(Node \*Temp)

{

if(Temp!=NULL)

{

inOrder(Temp->Left);

cout<<Temp->Data;

inOrder(Temp->Right);

}

}

void ExpTree::postOrder (Node \*Temp)

{

if(Temp!=NULL)

{

postOrder (Temp->Left);

postOrder(Temp->Right);

cout<<Temp->Data;

}

}

void ExpTree:: NonRpre()

{

Node \*Temp=Root,\*S[25];

int Top=-1;

while( Top >-1 || Temp!=NULL)

{

while(Temp!=NULL)

{

cout<<Temp->Data;

S[++Top]=Temp;

Temp=Temp->Left;

}

Temp=S[Top--];

Temp=Temp->Right;

}

}

void ExpTree::NonRin()

{

Node \*Temp=Root,\*S[25];

int Top=-1;

while( Top >-1 || Temp!=NULL)

{

while(Temp!=NULL)

{

S[++Top]=Temp;

Temp=Temp->Left;

}

Temp=S[Top--];

cout<<Temp->Data;

Temp=Temp->Right;

}

}

void ExpTree::NonRpost()

{

Node \*Temp=Root,\*S[25];

int I=0,Top=-1;

char Str[25];

while(Top>-1||Temp!=NULL)

{

while(Temp!=NULL)

{

Str[I++]=Temp->Data;

S[++Top]=Temp;

Temp=Temp->Right;

}

Temp=S[Top--];

Temp=Temp->Left;

}

Str[I]='\0';

for(I--;I>=0;I--)

cout<<Str[I];

}

int main()

{

ExpTree ob;

char exp[25];int ch;

cout<<"\nCreate tree 1:Postfix Expression 2:Prefix Expression";

cin>>ch;

if(ch==1)

{

cout<<"Enter Postfix Expression";cin>>exp;

ob.Post\_Create(exp);}

else

{

cout<<"Enter Prefix Expression";cin>>exp;

ob.Pre\_Create(exp);}

do

{

cout<<"\n1:Recursive 2:Non-Recursive 3:Exit";

cout<<"\nEnter your Choice: "; cin>>ch;

switch(ch)

{

case 1:

cout<<"Recursive=>";

cout<<"\nPreorder";

ob.preOrder(ob.return\_Root());

cout<<"\nInorder";

ob.inOrder (ob.return\_Root());

cout<<"\nPostorder";

ob.postOrder (ob.return\_Root());

break;

case 2:

cout<<"NonRecursive=>";

cout<<"\nPreorder";

ob.NonRpre();

cout<<"\nInorder";

ob.NonRin();

cout<<"\nPostorder";

ob.NonRpost();

break;

}

}

while(ch<3);

}