//dsa 2 stack code

#include<iostream>

#include<ctype.h>

#include<string.h>

using namespace std;

class Stack

{

//Structure for Expression

struct Stk

{

float Operator;

Stk \*Next;

Stk(){ Next=NULL;}

};

Stk \*Top;

public:

Stack(){Top=NULL;}

int Empty();

void Push(float Opr);

float Pop();

};

int Stack::Empty()

{

if(Top==NULL)

return 1;

return 0;

}

void Stack::Push(float Opr)

{

Stk \*Node;

Node=new Stk;

Node->Operator=Opr;

Node->Next=Top;

Top=Node;

}

float Stack::Pop()

{

Stk \*Temp=Top;

float Opr;

Top=Top->Next;

Opr=Temp->Operator;

delete Temp;

return Opr;

}

//Stack class End

//Function return Operater Priority

int Priority(char Op)

{

if(Op=='^')

return 2;

if(Op=='+' || Op=='-')

return 0;

else return 1;

}

//Return the result of given operation

float Operation(char Op,float A,float B)

{

int I=0;

float P=1;

if(Op=='\*') P=A\*B;

else if(Op=='/') P=A/B;

else if(Op=='+') P=A+B;

else if(Op=='-') P=A-B;

else while(I++<B) P=P\*A;

return P;

}

void infixTOpostfix(char str[20])

{

char Opr,post[20];

int i,j=0;

Stack S;

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

{

if(isalnum(str[i])) post[j++]=str[i];

else

{

if(str[i]== ')')

{

Opr=S.Pop();

while(Opr!='(')

{ post[j++]=Opr; Opr=S.Pop(); }//while

}

else { if(str[i]=='(');

else while(!S.Empty())

{

Opr=S.Pop();

if(Opr!='('&&Priority(Opr)>= Priority(str[i]))

post[j++]=Opr;

else

{S.Push(Opr);

break;}

}//while

S.Push(str[i]);

}

}

}//for

while(!S.Empty())

post[j++]=S.Pop();

post[j]='\0';

cout<<post;

}

void infixTOprefix(char str[20])

{

char Opr,pre[20];

int i,j=0;

Stack S;

for(i=strlen(str)-1;i>=0;i--)

{

if(isalnum(str[i])) pre[j++]=str[i];

else

{

if(str[i]== '(')

{

Opr=S.Pop();

while(Opr!=')')

{ pre[j++]=Opr; Opr=S.Pop(); }//while

}

else { if(str[i]==')');

else while(!S.Empty())

{

Opr=S.Pop();

if(Opr!=')'&&Priority(Opr)>Priority(str[i]))

pre[j++]=Opr;

else

{S.Push(Opr);

break;}

}//while

S.Push(str[i]);

}

}

}//for

while(!S.Empty())

pre[j++]=S.Pop();

pre[j]='\0';

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

cout<<pre[j];

}

float Postfix\_Evaluation(char String[20])

{

int I=0;

float Operand1,Operand2,Result;

Stack S;

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

{

if(String[I]>='0' &&String[I]<='9')

S.Push(String[I]-48);

else

{

Operand2=S.Pop();

Operand1=S.Pop();

Result=Operation(String[I],Operand1,Operand2);

S.Push(Result);

}

I++;

}

return S.Pop();

}

//PreFix Expression Evaluation

float Prefix\_Evaluation(char String[20])

{

int I=strlen(String)-1;

float Operand1,Operand2,Result;

Stack S;

while(I>=0)

{

if(String[I]>='0' &&String[I]<='9')

S.Push(String[I]-48);

else

{

Operand1=S.Pop();

Operand2=S.Pop();

Result=Operation(String[I],Operand1,Operand2);

S.Push(Result);

}

I--;

}

return S.Pop();

}

int main()

{

int Choice;

char Expression[25],Answer;

do

{

cout<<"\n1:Infix to Prefix\n2:Infix to Postfix\n3:PostfixEvaluation\n4:Prefix evaluation ";

cout<<"\nEnter your Choice: ";

cin>>Choice;

switch(Choice)

{

case 1:

cout<<"\nEnter infix Expression";

cin>>Expression;

infixTOprefix(Expression);

break;

case 2:

cout<<"\nEnter infix Expression";

cin>>Expression;

infixTOpostfix(Expression);

break;

case 3:

cout<<"\nEnter Postfix Expression";

cin>>Expression;

cout<<"\nEvaluated Result :"

<<Postfix\_Evaluation(Expression);

break;

case 4:

cout<<"\nEnter Prefix Expression";

cin>>Expression;

cout<<"\nEvaluated Result "

<<Prefix\_Evaluation(Expression);

break;

}

cout<<"\nContinue(y/n)...";

cin>>Answer;

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

return 0;

}