#include<iostream>

#include<string>

#include<ctype.h>

#include<string.h>

#include<algorithm>

using namespace std;

struct stack1

{

char data;

stack1\*next;

}\*top=NULL;

struct stack2

{

int data;

stack2\*next;

}\*top1=NULL;

int precedence (char op)

{

if (op=='^')

{

return 3;

}

else if(op='\*'||op=='/')

{

return 2;

}

else if(op='+'||op=='-')

{

return 1;

}

else

{

return 0;

}

}

void infixtopost(char infix[30])

{

stack1 \*newnode, \*temp;

string p;

int i=0,l;

newnode= new stack1;

newnode->data='(';

newnode->next=top;

top=newnode;

i=strlen(infix);

infix[1]=')';

while(top!=NULL)

{

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

{

newnode = new stack1;

newnode->data='(';

newnode->next=top;

top=newnode;

}

else if(infix[i]=='\*'|| infix[i]=='-'|| infix[i]=='+' || infix[i]=='/' ||infix[i]=='^')

{

while(precedence (top->data)>=precedence(infix[i]))

{

p+=top->data;

temp=top;

top=top->next;

delete temp;

}

newnode= new stack1;

newnode->data=infix[i];

newnode->next=top;

top=newnode;

}

else if(infix[i]==')')

{

while(top->data!='(')

{

p+=top->data;

temp=top;

top=top->next;

delete temp;

}

temp=top;

top=top->next;

delete temp;

}

else

{

p+=infix[i];

}

i++;

}

cout<<"\n Postfix expression is: "<<p;

}

void infixtopre(char infix[50])

{

stack1 \*newnode, \*temp;

string p;

int i,l;

i=strlen(infix)-1;

newnode= new stack1;

newnode->data=')';

newnode->next=top;

top=newnode;

infix[-1]='(';

while(top!= NULL)

{

if(infix[i]=')')

{

newnode=new stack1;

newnode-> data=')';

newnode-> next= top;

top =newnode;

}

else if(infix[i]=='\*'|| infix[i]=='-'|| infix[i]=='+' || infix[i]=='/' ||infix[i]=='^')

{

while(precedence (top->data)>=precedence(infix[i]))

{

p+=top->data;

temp=top;

top=top->next;

delete temp;

}

newnode= new stack1;

newnode->data=infix[i];

newnode->next=top;

top=newnode;

}

else if(infix[i]=='(')

{

while(top->data!=')')

{

p+=top->data;

temp=top;

top=top->next;

delete temp;

}

temp=top;

top=top->next;

delete temp;

}

else

{

p+=infix[i];

}

i--;

}

cout<<"\n Prefix expression is: ";

reverse(p.begin(),p.end());

cout<<p;

}

void evalpost(char p[30])

{

stack2 \*newnode, \*temp;

int i=0,a,b,c;

int l=strlen(p);

p[1]=')';

while(p[i]!=')')

{

if(isdigit(p[i]))

{

newnode =new stack2;

newnode-> data=p[i]- '0';

newnode-> next=top1;

top1=newnode;

}

else

{

a=top1->data;

temp= top1;

top1=top1->next;

delete temp;

b=top1->data;

temp=top1;

top1= top1->next;

delete temp;

switch(p[i])

{

case '+':

c=b+a;

break;

case '-':

c=b-a;

break;

case '\*':

c=b\*a;

break;

case '/':

c=b/a;

break;

}

newnode= new stack2;

newnode->data=c;

newnode->next=top1;

top1=newnode;

}

i++;

}

cout<<"\n value of postfix expression is: "<<top1->data;

}

void evalpre(char p[38])

{

stack2 \*newnode, \*temp;

int i=strlen(p)-1,a,b,c;

p[-1]='(';

while(p[i]!='(')

{

if(isdigit(p[i]))

{

newnode=new stack2;

newnode->data=p[i]-'0';

newnode->next=top1;

top1 =newnode;

}

else

{

a=top1->data;

temp=top1;

top1=top1->next;

delete temp;

b=top1->data;

temp=top1;

top1=top1->next;

delete temp;

switch(p[i])

{

case '+':

c=a+b;

break;

case '-':

c=a-b;

break;

case '\*':

c=a\*b;

break;

case '/':

c=a/b;

break;

}

newnode= new stack2;

newnode->data=c;

newnode->next=top1;

top1=newnode;

}

i--;

}

cout<<"\n value of prefix expression is: "<<top1->data;

}

int main()

{

char infix[30],p[30];

int ch;

do

{

cout<<"\n1. Infix to postfix";

cout<<"\n2. Infix to prefix";

cout<<"\n3. Evaluation of postfix";

cout<<"\n4. Evaluation of prefix";

cout<<"\n5. Exit";

cout<<"\n enter your choice:";

cin>>ch;

switch(ch)

{

case 1: cout<<"\n enter infix expression";

cin>>infix;

infixtopost(infix);

break;

case 2: cout<<"\n enter infix expression";

cin>>infix;

infixtopre(infix);

break;

case 3: cout<<"\n enter postfix expression:";

cin>>p;

evalpost(p);

break;

case 4: cout<<"\n enter prefix expression:";

cin>>p;

evalpre(p);

break;

}

}while(ch!=5);

return 0;

}

OUTPUT:

1. Infix to postfix

2. Infix to prefix

3. Evaluation of postfix

4. Evaluation of prefix

5. Exit

enter your choice:3

enter postfix expression:3,4,+,8,\*,2,3,5,/

value of postfix expression is: 3