```cpp
#include<iostream>
#include <string>
#include <stdio.h>
#include<math.h>
#include<iomanip>
using namespace std;
const int maxs = 8;  // 定义为偶数个
typedef double datatype;
// 操作数结点定义
typedef struct nodeint
{
    datatype data[maxs]={0};
    struct nodeint *next;
} StackNodeint;

// 运算符结点定义
typedef struct nodeoper
{
    char data;
    struct nodeoper *next;
}StackNodeoper;

// 操作数链栈定义
typedef struct
{
    StackNodeint *Top;
} LinkStackint;

// 运算符链栈定义
typedef struct
{
    StackNodeoper *Top;
}LinkStackoper;

// 建立空操作数链栈
void InitStackint(LinkStackint *&S)
{
    S = NULL;
    S = new LinkStackint;
    S->Top = NULL;
}

// 建立空运算符链栈
void InitStackoper(LinkStackoper *&S)
```

```
{
    S = NULL;
    S = new LinkStackoper;
    S->Top = NULL;
}

// 操作数进栈
void Pushint(LinkStackint *S1, StackNodeint *N)
{
    N->next = S1->Top;
    S1->Top = N;
}

//  运算符进栈
void Pushoper(LinkStackoper *S2,StackNodeoper *P)
{
    P->next = S2->Top;
    S2->Top = P;
}

// 操作数出栈
double Popint(LinkStackint *S1)
{
    StackNodeint *temp=S1->Top;
    double a=0,b=0; // 整数小数部分
    // 整数部分
    for (int i=maxs/2-1;i>=0;i--)
    {
        a=a+temp->data[i]*pow(10.0,maxs/2-i-1);
    }
    // 小数部分
    for (int m=maxs/2;m<maxs-1;m++)
    {
        b=b+temp->data[m]*pow(10.0,(-1*(m+1-maxs/2)));
    }
    a=a+b;
    S1->Top=temp->next;
    delete temp;
    return a;
}

// 运算符出栈
char Popoper(LinkStackoper *S2)
{
```

```
    char c;
    StackNodeoper *temp=S2->Top;
    S2->Top=temp->next;
    c=temp->data;
    delete temp;
    return c;
}

// 字符串检查函数，测试表达式是否有效，若有效则返回 1，无效则返回 0
int check (string &s)
{
    int len=s.length();
    int m=0,n=0;
    for(int j=0;j<len;)
    {
        if(s[j]==s[j+1]&&s[j+1]=='+')
        // 连续为加号时则删除多余加号
            {
                s.erase(j,1);
            }
        else if(s[j]=='(')
            {   // 括号内为空
                if(j<len-1&&s[j+1]==')')
                    {
                        cout<<"输入的表达式无效：括号不匹配"<<endl;
                        return 0;
                    }
                else
                {
                    m=m+1;
                    j=j+1;
                }
            }
        else if(s[j]==')')
            {
                // 右括号比左括号多
                if(m==0)
                    {
                        cout<<"输入的表达式无效：括号不匹配"<<endl;
                        return 0;
                    }
                else
                    {
                        m=m-1;
```

```
                        j=j+1;
                    }
            }
        // 运算符混合输入
        else if(s[j]=='*'||s[j]=='+'||s[j]=='-'||s[j]=='/')
            {
                if(s[j-1]=='*'||s[j-1]=='+'||s[j-1]=='-'||s[j-1]=='/')
                    {
                        cout<<"输入的表达式无效"<<endl;
                        return 0;
                    }
                else j=j+1;
            }
        else j=j+1;
    }
    // 若左括号比右括号多
    if(m!=0)
        {
            cout<<"输入的表达式无效：括号不匹配"<<endl;
            return 0;
        }
    else return 1;
}

// 四则运算函数
double calc(string s)
{   LinkStackint *S1;   // 操作数链栈
    LinkStackoper *S2;  // 运算符链栈
    InitStackint(S1);InitStackoper(S2);     // 链栈初始化
    StackNodeint *p,*p1;    // 操作数
    StackNodeoper *r;   // 运算符
    double pop1,pop2,pop12;
    int j=0;
    // 考虑第一个数是运算符 "-" 或 "+"
    p=NULL;
    p=new StackNodeint;
    p->data[maxs/2-1]=0;
    Pushint(S1,p);
    // 字符串检查
    int chec=check(s);
    if(chec==0)
        return 3.14159;
    else NULL;
    // 检查后的字符串加括号
```

```
s='('+s+')';
// 最终字符串长度
int a = s.length();
for (j=0;j<=a-1;)
    {
        int n=1,m=0;
        // 若为数字则读入数据并计算（可能）
        if (int(s[j])>=48&&int(s[j])<=57)
            {   p=NULL;
                p=new StackNodeint;
                p->data[maxs/2-1]=int(s[j])-48;
                n=n+1;
                j=j+1;
                while(1)  //  是否继续读入数字
                {
                    // 若仍为数字
                    if(int(s[j])>=48&&int(s[j])<=57)
                        {
                        for(int i=n;i>1;i--)
                            // 数字移动
                            p->data[maxs/2-i]=p->data[maxs/2-i+1];
                        p->data[maxs/2-1]=int(s[j])-48;
                        j=j+1;
                        n=n+1;
                        // 若 j是最后一个字符
                        if(j>a-1)
                            break;
                        }
                    // 跳过小数点
                    else if(s[j]=='.')
                        {   j++;
                            m=0;
                            // 读取小数部分
                            while(1)
                            {
                                p->data[maxs/2+m]=int(s[j])-48;
                                m=m+1;j=j+1;
                                if(j>a-1)
                                    break;
                                if(int(s[j])<48||int(s[j])>57)
                                    break;          // 跳出循环
                            }
                        break;                  // 跳出外循环
                        }
```

```
            // 后面无数字则 break
            else break;
        }
        //  操作数进栈
        Pushint(S1,p);
        // 计算可以计算的操作数
        // 避免刚开始运算符栈为空
        if (S2->Top==NULL)
        {
            NULL;
        }
        // 若之前运算为乘方则计算
        else if(S2->Top->data=='^')
        {
            Popoper(S2);
            pop1=Popint(S1);
            pop2=Popint(S1);
            pop12=pow(pop2,pop1);
            p=NULL;
            p=new StackNodeint;
            p->data[maxs/2-1]=pop12;
            Pushint(S1,p);
            // 若乘方之后还是乘方
             if(s[j]=='^')
                NULL;
            // 乘方后无乘方则处理之前的乘除
            else if(S2->Top->data=='*'||S2->Top->data=='/')
            {
                pop1=Popint(S1);
                pop2=Popint(S1);
                if(S2->Top->data=='*')
                    pop12=pop2*pop1;
                else
                    pop12=pop2/pop1;
                p=NULL;
                p=new StackNodeint;
                p->data[maxs/2-1]=pop12;
                Pushint(S1,p);
                Popoper(S2);
            }
            else NULL;
        }
        // 若为乘法则计算
        else if(S2->Top->data=='*'&& (s[j]!='^'))
```

```
                    {
                        Popoper(S2);
                        pop1=Popint(S1);
                        pop2=Popint(S1);
                        pop12=pop2*pop1;
                        p1=NULL;
                        p1=new StackNodeint;
                        p1->data[maxs/2-1]=pop12;
                        Pushint(S1,p1);
                    }
                // 若为除法也计算
                else if(S2->Top->data=='/'&&s[j]!='^')
                {
                    Popoper(S2);
                    pop1=Popint(S1);
                    pop2=Popint(S1);
                    pop12=pop2/pop1;
                    p1=NULL;
                    p1=new StackNodeint;
                    p1->data[maxs/2-1]=pop12;
                    Pushint(S1,p1);
                }
                // 其他不计算
                else
                    {NULL;}
            }
        // 不为数据则运算符进栈
        else
            {   r=NULL;
                r=new StackNodeoper;
                r->data=s[j];
                Pushoper(S2,r);
                // 考虑括号内第一个为负数
                if(s[j]=='(')
                {
                    if(s[j+1]=='-'||s[j+1]=='+')
                        {
                            p=NULL;
                            p=new StackNodeint;
                            p->data[maxs/2-1]=0;
                            Pushint(S1,p);
                        }
                    else
                        NULL;
```

```
                    }
                    // 遇到 ")"时，计算括号内的加减法
                    else if(s[j]==')')
                    {
                        Popoper(S2);
                        // 循环直到将括号去掉
                        while(1)
                        {
                            // 括号去掉则 break
                            if(S2->Top->data=='(')
                            {
                                Popoper(S2);
                                break;
                            }
                            pop1=Popint(S1);
                            pop2=Popint(S1);
                            // 计算括号内的加减法
                            if(S2->Top->data=='+')
                                {   if(S2->Top->next->data=='-
'&&S2->Top->next->data!='(')
                                    {
                                        pop12=pop2-pop1;
                                    }
                                    else
                                        pop12=pop2+pop1;
                                }
                            else  // 为减号
                                {   if(S2->Top->next->data=='-
'&&S2->Top->next->data!='(')
                                        pop12=pop2+pop1;
                                    else
                                        pop12=pop2-pop1;
                                }
                            p1=NULL;
                            p1=new StackNodeint;
                            p1->data[maxs/2-1]=pop12;
                            Pushint(S1,p1);
                            Popoper(S2); // 计算完成后运算符出栈
                            if(S2->Top->data=='(')
                            {
                                Popoper(S2);
                                break;
                            }
                        }
```

```
                    // 避免第一个是括号
                    if(S2->Top==NULL)
                    {
                        NULL;
                    }
                    // 括号前是乘方则计算
                    else if(S2->Top->data=='^')
                    {
                        Popoper(S2);
                        pop1=Popint(S1);
                        pop2=Popint(S1);
                        pop12=pow(pop2,pop1);
                        p=NULL;
                        p=new StackNodeint;
                        p->data[maxs/2-1]=pop12;
                        Pushint(S1,p);
                        // 考虑乘方计算后的前后
                        if(s[j+1]=='^')
                            NULL;
                        // 后面无乘方则计算乘除
                        else if(S2->Top->data=='*'||S2->Top->data=='/')
                        {
                                pop1=Popint(S1);
                                pop2=Popint(S1);
                                if(S2->Top->data=='*')
                                    {
                                        pop12=pop2*pop1;
                                    }
                                else
                                    pop12=pop2/pop1;
                                p1=NULL;
                                p1=new StackNodeint;
                                p1->data[maxs/2-1]=pop12;
                                Pushint(S1,p1);
                                Popoper(S2);
                        }
                        // 加减运算不做任何操作
                        else
                            {NULL;}
                    }
                    // 下一个运算是乘方则不计算
                    else if(s[j+1]=='^')
                        NULL;
                    // 如已去掉的括号前面是 乘除法则计算
```

```
                        else if(S2->Top->data=='*'||S2->Top->data=='/')
                        {
                            pop1=Popint(S1);
                            pop2=Popint(S1);
                            if(S2->Top->data=='*')
                                {
                                    pop12=pop2*pop1;
                                }
                            else
                                pop12=pop2/pop1;
                            p1=NULL;
                            p1=new StackNodeint;
                            p1->data[maxs/2-1]=pop12;
                            Pushint(S1,p1);
                            Popoper(S2);
                        }
                        // 其他运算不做任何操作
                        else
                            {NULL;}
                    }
                    // 否则忽略 （没遇到括号）
                    else {NULL;}
                    j=j+1;
                }
        }
    pop1=Popint(S1);
    // 若操作数栈顶的 0 没有参与运算则删除该结点
    if(S1->Top!=NULL)
        pop2=Popint(S1);
    return pop1;
}
int main()
{
string str;
getline(cin,str);
double c=calc(str);
cout<<c<<endl;
return 0;
}
```