实验报告 1

姓名：章东泉　　学号：PB17121706

问题 1：

结果如下图所示。

```
x:0.125000000000e-000   f(x):0.260303728282e-002        g(x):0.260303728282e-002
x:0.156250000000e-001   f(x):0.406898288929e-004        g(x):0.406898288929e-004
x:0.195312500000e-002   f(x):0.635782782865e-006        g(x):0.635782782865e-006
x:0.244140625000e-003   f(x):0.993410687045e-008        g(x):0.993410775862e-008
x:0.305175781250e-004   f(x):0.155220281073e-009        g(x):0.155220433729e-009
x:0.381469726563e-005   f(x):0.242517117499e-011        g(x):0.242531927701e-011
x:0.476837158203e-006   f(x):0.377475828373e-013        g(x):0.378956137032e-013
x:0.596046447754e-007   f(x):0.444089209850e-015        g(x):0.592118964113e-015
x:0.745058059692e-008   f(x):0.000000000000e-000        g(x):0.925185881427e-017
x:0.931322574615e-009   f(x):0.000000000000e-000        g(x):0.144560293973e-018
```

显然，用 $g(x)=\dfrac{x^2}{\sqrt{x^2+9}+3}$ 计算得到的结果更精确。

由于实验数据都非常小，所以利用 $f(x)=\sqrt{x^2+9}-3$ 计算时会出现两个相近的数相减的情况，会增大相对误差。容易看出，当输入 x 比较大时 $f(x)$ 与 $g(x)$ 结果相差不大，但随着 x 的减小， $f(x)$ 的误差越来越大。综上， $g(x)$ 的结果更可靠。

实验计算采用 C 语言，直接使用 *math.h* 库函数进行计算，函数如下：

```c
float function_f(float x)
{
    float result = sqrt(pow(x, 2) + 9) - 3;
    return result;
}

float function_g(float x)
{
    float result = pow(x, 2)/(sqrt(pow(x, 2) + 9) + 3);
    return result;
}
```

得到的结果使用 sprintf 函数将科学记数法格式的结果写入到字符串中，经过简单处理后输出。

注：这里是因为标准%.12e 输出的科学记数法从第一位有效数字

开始，与参考数据不符故应进行转换。

问题 2：

结果如下图所示：

```
Method1:1.025188e-010   Method2:-1.564331e-010  Method3:0.000000e+000
```

第二种算法相对精确。如果采用第一种算法，前三个数的和是-2755462.87406667921304，与第四个数的绝对值过于接近，相加时会使相对误差变大。如果采用第三种算法，正数和负数的和分别是：

```
positive: 2.759502919618e+006   negative: -2.759502919618e+006
```

同样是绝对值很接近的数。而使用第二种方法则有效地避免了这样的问题。

这道题的代码结构过于简单，故不做分析。

```
double x1 = 4040.045551380452;
double x2 = -2759471.276702747;
double x3 = -31.64291531266504;
double x4 = 2755462.874010974;
double x5 = 0.0000557052996742893;

int main()
{
    double result = 0.0;
// calculation 1
    result += x1;
    result += x2;
    result += x3;
    result += x4;
    result += x5;
    printf("Method1:%.6e\t", result);
// calculation 2
    result = 0.0;
    result += x5;
    result += x4;
    result += x3;
    result += x2;
    result += x1;
    printf("Method2:%.6e\t", result);
//calculation 3
    double part1 = 0.0;
    part1 += x4;
    part1 += x1;
    part1 += x5;
    double part2 = 0.0;
    part2 += x2;
    part2 += x3;
    result = part1 + part2;
    printf("Method3:%.6e\n", result);
    printf("positive: %.12e\tnegative: %.12e\n",x4+x1+x5, x2+x3);
    return 0;
}
```