Numerical Analysis Programming Assignment #1

姓名：陈宇轩 学号：PB16060738

问题1

编写程序分别计算$f(x)=\sqrt{x^2+4}-2$和$g(x)=\frac{x^2}{\sqrt{x^2+4}+2}$分别令 $x=8^{-1},\ 8^{-2},\ 8^{-3},\ \ldots,\ 8^{-10}$，用单精度（即float型变量）进行计算，输出所有的计算结果，结果保留12位尾数(用科学计数形式)，比较并分析两种方法得到的计算结果。你认为哪种方法得到的计算结果更可靠？请给出你的理由或分析。

计算结果

x	$(x^2+4)^{0.5}-2$	$\frac{x^2}{\sqrt{x^2+4}+2}$
1.250000000000e-01	3.902442753315e-03	3.902442753315e-03
1.562500000000e-02	6.103422492743e-05	6.103422492743e-05
1.953125000000e-03	9.536740890326e-07	9.536740890326e-07
2.441406250000e-04	0.000000000000e+00	1.490116119385e-08
3.051757812500e-05	0.000000000000e+00	2.328306436539e-10
3.814697265625e-06	0.000000000000e+00	3.637978807092e-12
4.768371582031e-07	0.000000000000e+00	5.684341886081e-14
5.960464477539e-08	0.000000000000e+00	8.881784197001e-16
7.450580596924e-09	0.000000000000e+00	1.387778780781e-17
9.313225746155e-10	0.000000000000e+00	2.168404344971e-19

结果分析

我认为**第二种**方法得到的结果更可靠。因为在计算中，要尽量避免两相近数相减，否则会导致相对误差变大，有效数字丢失，发生减性抵消。而第二种方法避免了这一问题，则显示了更为可靠的结果。

问题2

求向量X=[4040.045551380452, -2759471.276702747, -31.64291531266504, 2755462.874010974, 0.0000557052996742893]的和，分别采取以下3种方式：（a) 顺序求和；（b) 逆序求和；（c) 正数从大到小求和，负数从小到大求和，再相加； 用双精度进行计算，计算结果保留7位有效数字（用科学计数形式，比如1.234567E-10）. 比较3种方法得到的计算结果；你认为哪种方法得到的计算结果更精确？试给出你的理由或分析。

计算结果

	方法(a)	方法(b)	方法(c)
计算结果	1.025188e-10	-1.564331e-10	0.000000e+00

结果分析

我认为，**方法(a)**得到的计算结果更精确。首先对于方法(c)，正数从大到小求和，在计算机处理大数加小数时，需要对阶处理，会出现大数+小数≈大数。因此，方法(c)误差最大。对于(a)和(b)，主要区别在于方法(b)中首先将小数0.00005570529967428 93和大数2755462.874010974相加，则此小数的影响很小，导致误差变大。而方法(a)在最后加上0.00005570529967428 93时，两数接近，使得误差更小一点，因此结果也更精确。

总结

在编程过程中，虽然单精度和双精度在计算机中的表示格式一样，但由于位数不同，需要注意数值的传递中是否发生了类型的转换，否则容易出问题。

计算过程中，为了避免误差危害，需尽量遵循以下规则：

1. 选择数值稳定的计算方法，避免两相近数相减。
2. 合理安排计算顺序，防止大数"淹没"小数。

实验代码（Matlab实现）

```
%问题1
x = 1;
x = single(x);
for i=1:10
    x = x/8;
    fprintf("x:%.12e ",x);
    fx = sqrt(x^2+4)-2;
    fprintf("f(x):%.12e  ",fx);
    gx = x^2/(sqrt(x^2+4)+2);
    fprintf("g(x):%.12e\n",gx);
end

%问题2
sum1 = 0; sum2 = 0;
x = [4040.045551380452, -2759471.276702747, -31.64291531266504,  2755462.874010974,
0.00005570529967428 93];
for i=1:5
    sum1 = sum1 + x(i);
    sum2 = sum2 + x(6-i);
end
fprintf("(a)顺序求和结果: %7e\n", sum1);
fprintf("(b)逆序求和结果: %7e\n", sum2);
sum1 = x(4) + x(1) + x(5);
sum2 = x(3) + x(2);
sum1 = sum1 + sum2;
fprintf("(c)正数从大到小求和，负数从小到大求和，再相加求和结果: %7e\n", sum1);
```