问题 1.编写程序分别计算

$$f(x)=\sqrt{x^2+4}-2 \text{ 和 } g(x)=\frac{x^2}{\sqrt{x^2+4}+2}\text{；}$$

分别令 x=8^{-1}，8^{-2}，8^{-3}，…，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。

1.计算方法与结果

用 C 编写程序进行计算，可得计算结果如下表所示：

表 1

x	(x^2+4)^0.5-2	(x^2)/((x^2+4)^0.5+2)
0.125000000000E-000	0. 390244275331E-002	0. 390244275331E-002
0.156250000000E-001	0. 610342249274E-004	0. 610342249274E-004
0.195312500000E-002	0. 953674089033E-006	0. 953674089033E-006
0.244140625000E-003	0. 000000000000E-000	0. 149011611938E-007
0. 305175780000E-004	0. 000000000000E-000	0. 232830643654E-009
0. 381469700000E-005	0. 000000000000E-000	0. 363797880709E-011
0. 476837000000E-006	0. 000000000000E-000	0. 568434188608E-013
0. 596050000000E-007	0. 000000000000E-000	0. 888178419700E-015
0. 745100000000E-008	0. 000000000000E-000	0. 138777878078E-016
0. 931000000000E-009	0. 000000000000E-000	0. 216840434497E-018

注：表中“x”一列从上到下依次是 8^{-1},8^{-2}，…

2.结果分析

- 从结果中可以看到，运用方法一计算，从第 4 行往后，计算所得值都为 0。运用方法二计算，每一行都有一个非零结果值。
- 产生这种情况的原因是,方法一中存在着**相近数相减**的问题,当 x 值很小时，float 型变量的储存精度有限，此时 x 在电脑中储存为 0，故$\sqrt{x^2+4}$与 2差值即为 0，这便是为什么从第 4 行往后，计算所得值都为 0。
- 由以上分析，方法二的计算结果更可靠，因为避免了**相近数相减**的问题，减少了精度的损失。

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 ***1.234567*E-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。**

1.计算方法与结果

表 2

	方法(a)	方法(b)	方法(c)
计算结果	1.025188E-010	-1.564331E-010	0.000000E+000

2.结果分析

- 人工计算得到准确值为 8.66342893E-11，三种方法的绝对误差如下：

	绝对误差
方法(a)	-0.158845107E-10
方法(b)	2.430673893E-10
方法(c)	0.66342893E-10

可见方法(a)与精确值误差最小，故方法 a 的计算结果最精确

- 引起计算误差的原因是在浮点数可以表示的位数有限，在加减运算中如果中间结果有效位数过多，则会造成精度的丢失。同时，十进制值通常没有完全相同的二进制表示形式，为此，可能会经历一些精度丢失。
- 下面分析 3 种算法在每一步中，机器计算和人工计算的差别

方法(a)

【步骤一】t0=x0+x

人算	t0	-2.7554312311513665**48**e+006	小数点后第几位发生偏差
机算	t0	-2.7554312311513666**00**e+006	10

【步骤二】t1=t0+x2

人算	t1	-2.7554628740666792**6504**e+006	小数点后第几位发生偏差
机算	t1	-2.7554628740666793**0000**e+006	10

【步骤三】t2=t1+x3

人算	t2	-5.5705**3**e-005	小数点后第几位发生偏差
机算	t2	-5.5705**197155475616**e-005	10

【步骤四】t3=t2+x4

人算	T3	1.0251881369**4**e-010	小数点后第几位发生偏差
机算	T3	1.0251881368**296672**e-010	20

方法(b)

【步骤一】t0=x3+x4

人算	t0	2.755462874066679**2996742893**e+006	小数点后第几位发生偏差
机算	t0	2.755462874066679**30000**e+006	10

【步骤二】t1=t0+x2

人算	t1	2.755431231151366634 69e+006	小数点后第几位发生偏差
机算	t1	2.755431231151366600 00e+006	11

【步骤三】t2=t1+x1

人算	t2	-4.040045551380 4e+003	小数点后第几位发生偏差
机算	t2	-4.040045551380 6082e+003	10

【步骤四】t3=t2+x0

人算	T3	-1.562e-010	小数点后第几位发生偏差
机算	T3	-1.564330887049436 6e-010	13

方法(c)

【步骤一】t0=x0+x3

人算	t0	2.759502919562354 452e+006	小数点后第几位发生偏差
机算	t0	2.759502919562354 700e+006	10

【步骤二】t1=t0+x4

人算	t1	2.759502919618059 9996742893e+006	小数点后第几位发生偏差
机算	t1	2.759502919618059 90000e+006	11

【步骤三】t2=x1+x2

人算	t2	-2.759502919618059 66504e+006	小数点后第几位发生偏差
机算	t2	-2.759502919618059 90000e+006	10

【步骤四】t3=t1+t2

人算	T3	0	小数点后第几位发生偏差
机算	T3	0.000000000000000000 00e+000	本步无偏差

根据三种方法在每一个计算步骤中发生的偏差，我们可以发现，在计算过程中，中间结果的有效位数越多，丢失的精度越多，具体表现为方法(c)中，中间结果的整数部分位数过多，明显造成了小数部分的精度丢失。

【小结】

为何浮点数可能丢失精度浮点十进制值通常没有完全相同的二进制表示形式。 这是 CPU 所采用的浮点数据表示形式的副作用。 为此，可能会经历一些精度丢失，并且一些浮点运算可能会产生意外的结果。

导致此行为的原因是下面之一：

1. 相近数相减会造成精度丢失，增大绝对误差。在计算过程中要尽量避免相近数相减。
2. CPU 采用的浮点数据表示形式的副作用是十进制数的二进制表示形式可能不精确。Float 和 double 类型的精度有限，数据的有效数字过多会造成精度的丢失。
3. 运用浮点数计算的过程中，要注意原数据和计算过程中的中间数据的精度，有效数字位数尽量不要超过 float 或 double 类型的表示范围。