C for Science - Practical Exercise #2

1. Create a quadratic solver in C for the following equation: $A[2]x^2 + A[1]x + A[0] = 0$

   (a) Create a new C project containing two `.c` files: `main.c` and `quadSol.c`.

   (b) Start with the following code for `main.c`:

```
#include <stdio.h>
int quad_roots(double * A, double * roots);

int main(void)
{
   double A[3], roots[2];

   printf("Enter coefficients of Quadratic Equation a2*x*x + a1*x + a0 = 0\n"
   "in the order a2, a1, a0, separated by spaces:");
   scanf("%lf %lf %lf", A+2, A+1, A);

   switch(quad_roots(A, roots))
   {
   case 0:
      printf("x = %f\nx = %f\n",roots[0], roots[1]);
      break;
   }
   return 0;
}
```

   (c) In `quadSol.c` create the function `quad_roots` as per the declaration in `main.c`. The function should return the following values (use logical expressions and control statements to establish which case):

   `-3` when `A[2] = A[1] = 0, A[0]` $\neq$ `0`, not an equation.
   `-2` when `A[0] = A[1] = A[2] = 0`, no meaningful solution.
   `-1` when `A[2] = 0, A[1]` $\neq$ `0`, a linear equation with one solution `roots[0]`.
   `0` for two different real roots (`A[1]`^{2} `> 4A[2]A[0]`), `roots[0]` and `roots[1]`.
   `1` for complex conjugate roots (`A[1]`^{2} `< 4A[2]A[0]`), `roots[0]` should contain the real part and `roots[1]` the imaginary part.
   `2` for the case with two equal real roots `A[1]`^{2} `= 4A[0]A[2]`, `roots[0] = roots[1]`.

   For reference the roots of the quadratic are given by: $x = \frac{-A[1] \pm \sqrt{A[1]^2 - 4A[2]A[0]}}{2A[2]}$
   HINT: The file might begin:

```
#include <math.h>
int quad_roots(double * A, double * roots)
{
   if (A[2] == 0)
   {
      if (A[1] == 0)
      {
         if (A[0] == 0)
         {
            //No meaningful sol
            return -2;
         }
         else
         ...
```

   (d) Expand the `switch` block in `main.c` to include appropriate `printf` lines for each of the possible 6 return values.

   (e) The program should be tested on the data set:

   i. `A[2] = 1, A[1] = 3, A[0] = 2,`
   ii. `A[2] = 1, A[1] = 2, A[0] = 3,`
   iii. `A[2] = 4, A[1] = -4, A[0] = 1,`
   iv. `A[2] = 0, A[1] = 2, A[0] = 1.`

2. The C Standard header file `<float.h>` contains information about the various floating point numbers used in the C system such as largest and smallest numbers (in magnitude). Standard C header files are installed with the compiler, in Windows look in `C:\Program Files (x86)\Microsoft Visual Studio 10.0\VC\include` or similar, and in Linux look in `/usr/local/include` or run `locate float.h`. One useful quantity is the smallest number that can be added to `1.0` and cause a difference to the value (this is known as ϵ).

   (a) Locate and open the `<float.h>` used by your compiler and extract the following constants `FLT_MAX`, `FLT_MIN` and `FLT_EPSILON`.

   (b) What are the corresponding values for `double` types?

   (c) Given the following doubles:

   ```
   double a = 1.0 + DBL_EPSILON;
   double b = (1.0 + DBL_EPSILON / 2.0) + DBL_EPSILON / 2.0;
   double c = 1.0 + (DBL_EPSILON / 2.0 + DBL_EPSILON / 2.0);
   ```

   Print out `1.0-a`, `1.0-b` and `1.0-c`, using the `%g` format specifier, is this what you expect? We see even the process of summing up numbers can produce problems. One famous algorithm to sum up numbers accurately is *Kahan Summation* (look this up!).

3. *Combatting rounding errors:* Returning to our quadratic solver, if `A[1]`$^2 \gg$ `4A[0]A[2] > 0`, then the calculation of one of the roots (`-A[1]` $\pm\sqrt{\cdot}$)/`2A[2]` involves a subtraction of nearly equal quantities, and thus a loss of accuracy. ('Adding' two numbers of similar magnitude and different sign also results in an accuracy loss!). When possible, an alternative formulation should be sought which avoids this subtraction.

   One useful property of quadratics is that the roots satisfy the following expression:

   ```
   roots[0]*roots[1] = A[0]/A[2]
   ```

   Thus a more accurate way to calculate both roots of the quadratic is to choose the sign $\pm$ such that (`-A[1]` $\pm\sqrt{\cdot}$)/`2A[2]` is not a subtraction, and then compute the other root from the equation above.

   (a) Edit your `main` function to print roots to 20 decimal places.

   (b) Solve the equation $x^2 + 450.0025x + 1.125 = 0$ using your quadratic solver. The roots should be $x = -450$ and $x = -0.0025$

   (c) Modify `quad_sol` to avoid subtraction when computing roots.

   (d) Compare the modified `quad_sol` to the original. Is there a difference?