C++: Practical session 2

1 Solution to a quadratic

Your task is to implement a robust program to compute the solution(s) to a quadratic equation entered by ther user. Your program should be able to deal with all possible cases that could arise.

Example run:

```
Solving a*x*x + b*x + c = 0:
Enter a: 2
Enter b: 5
Enter c: -3
Solutions are 0.5 and -3.
Enter a: 3
Enter b: 0
Enter c: 1
There are no real solutions
```

1.1 Extensions

a) Extend your program to print any complex solutions to the quadratics (Note the existence of `complex.h`. However, this *does not* deal well with floating-point overflow, and should be avoided when any robust system is required.)

b) (For masochists only): Write a similar version for cubics and/or quartics

2 Squares

Write a program that reads an integer from the user, and then proceeds to list all the squares from 0 up to that integer.
An example run would look like:

```
Please enter the maximum square: 4
0 * 0 = 0
1 * 1 = 1
2 * 2 = 4
3 * 3 = 9
4 * 4 = 16
```

2.1 Extensions

a) What happens if the user enters -4? Change your program to ask the user for another number if their input does not make sense.

b) Output the values to a file that can be plotted in `gnuplot`. The file format should be two columns, with x in the first column, and x^2 in the second.

   This can be plotted in `gnuplot` using `plot "squares.dat" using 1:2 with lines`

3 Type-limits

Use the templated class `std::numeric_limits<T>` to find out about as many of the built-in types in your C++ implementation as possible.

For integral types, you should be able to find out:

- The number of bits used
- The range of values held
- The signedness

For floating-point types, you should be able to find:

- The number of bits used for the exponent and mantissa
- The range of values held
- The range of the exponent
- The smallest value such that $1 + x > 1$.