Assignment 1: Results/Description

Problem 1

I've created a program `myquadratic_single` in C++ that solves a given quadratic equation. The coefficients `a`, `b`, and `c` corresponding to the equation are provided by the user in a parameter file. The program puts the equation into the form $x^2 + 2bx + c = 0$. It then performs the calculation of the solution in the standard way (i.e. with addition and subtraction of the discriminant term), as well as a more precise method that makes use of division in place of subtraction (subtraction is dangerous).

I tested it with a couple known cases. First, I tried the trivial equation $x^2 - 1 = 0$ (`a=1`, `b=0`, `c=-1`), and the output of the program is as expected:

```
Reading from parameter file: ./config/params
Attempting to solve equation 1x^2 + 0x + -1 = 0
Standard calculation:
-1  1
Accurate calculation:
-1  1
```

I also tested the equation $3.2x^2 + x - 6.5 = 0$, and the solutions are given to `double` precision:

```
Reading from parameter file: ./config/params
Attempting to solve equation 3.2x^2 + 1x + -6.5 = 0
Standard calculation:
-1.5900087183693077  1.2775087183693077
Accurate calculation:
-1.5900087183693077  1.2775087183693079
```

Problem 2

I've placed the quadratic solver in its own implementation/header ("quadratic.cpp/hpp"), and one may `include` this functionality as a subroutine, etc. in another file with an `#include "quadratic.hpp"`. This has been done in the second main implementation "myquadratic.cpp".

Problem 3

I have written the Makefile for this program, which compiles each object and links them to create the executable file. To compile this program, one may just run `make`. This creates the "myquadratic" executable, and it does the exact same thing as before, but it allows more flexibility in writing the code.

Problem 4

(a)

As a test for imaginary roots, I can just use the equation $x^2 + 1 = 0$. When doing this, in C++ the `std::sqrt` function returns `NaN`, so no result is found. However, an error is not given and the code continues to run, which could be dangerous or non-informative.

One may alter the code to handle these imaginary solutions in a few ways. The simplistic approach that I chose for this simple program was to incorporate an exception to be thrown if an imaginary solution arises. The output from my program to such an equation provided gives:

```
Reading from parameter file: ./config/params
Attempting to solve equation 1x^2 + 0x + 1 = 0
Error: Imaginary value(s) encountered in solution.
```

One may also incorporate complex number functionality if available. `complex` objects are available in `std::complex` for C++, for example. These of course require over double the amount of memory, though.

(b)

With the extreme equation $x^2 - 200000x + 1 = 0$, my program outputs the following:

```
Reading from parameter file: ./config/params
Attempting to solve equation 1x^2 + -200000x + 1 = 0
Standard calculation:
4.9999944167211652e-06  199999.99999500002
Accurate calculation:
199999.99999500002  5.0000000001249996e-06
```

One of the solutions is slightly different from the other (albeit still close to somewhat high precision). As was mentioned, the standard calculation uses subtraction, which has the

potential to destroy some significant precision at the cost of faster processing time. On the other hand, the accurate method only performs a division to get the second solution from the first, leading to a more precise yet slower result.