(Real) Root Finding

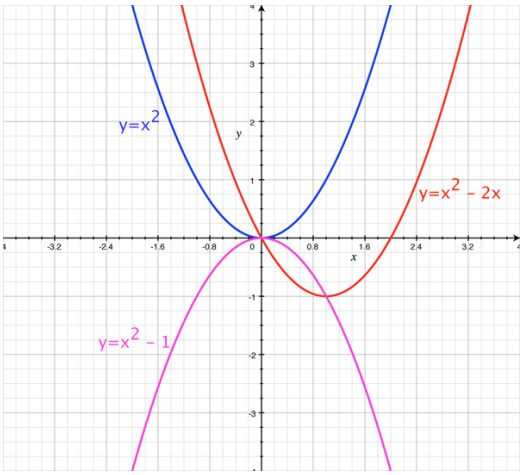
This is an exasmple of a program that on initial glance seems trivial to solve. But our experience has shown that most students submit an incorrect and/or incomplete solution. This initial assignment is intended to impress upon you the following quote:

> *"In computer science, if you are almost correct, you are a liability."*
> Fred Kollett (1941-1997), MathCS, Wheaton College, Norton, MA

A quadratic equation is a second-order polynomial equation in a single variable x

$$ax^2 + bx + c = 0, \qquad \textit{where } a \neq 0$$

Because it is a second-order polynomial equation, the fundamental theorem of algebra guarantees that it has two solutions. These solutions may be both real or both complex, although we will focus on finding Real roots here. Some sample graphs and **a**, **b**, **c** coefficients are shown here.



(blue): `a=1, b= 0, c= 0` *or* $y = x^2$
(red): `a=1, b=-2, c= 0` *or* $y = x^2 - 2x$
(pink): `a=1, b= 0, c=-1` *or* $y = x^2 - 1$
(hey wait; are these plots correct?)

Given the coefficients, the **quadratic equation** will give you the two roots (note the ± notation):

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

(0) From the Starter Kit folder, replace your Project's default main.cpp file with the **root.cpp** file (that is, remove main.cpp and add root.cpp file to your Project).

(1) As a start, fix the **roots.cpp** code so that it will **trap negative numbers *before* taking the square root**. Thus, your code will *not* find complex roots; it is your job to trap situations that might involve complex roots and print appropriate WARNING messages. That is, do not call `sqrt()` if the **discriminant ($b^2 - 4ac$)** is negative. If the discriminant is negative, you *must* print an appropriate message (what *is* an appropriate message? You do know that it is ok to seek online helps and discussions of this topic, right? But, be wise, the assignment isn't to go out and *find* a solution in C++, rather this assignment represents practice for you to consider multiple cases and write correct solutions. That said, yes, please do read up on what it means for the discriminant to be negative and print a good message in this case).

(2) Change the program so that it **prompts the user** to enter (real) coefficients for **`a`**, **`b`**, and **`c`**. This way, you won't have to keep recompiling just to change them, right? Note: but, of course, it is best to just test your code using "burned in" values; you do not want to spend time typing in your values for A, B, and C *while* you are just testing your code. (So make this change in later stages of your development).

(3) Your program must trap all "bad" cases. What *could* happen? What are the possible combination of values for the three coefficients? Draw various types of graphs and consider the coefficients that lead to such graphs.

(4) ***Note: your program must handle quadratic equations with two Real roots (and print those two values), double roots, linear equations (lines) and print the one Real root, as well as trap bad cases.*** When programmers need to be sure they have *exhaustively* tested their programs, they often

create what is called a **test suite**. In fact, on big projects, the Quality Assurance (QA) group is responsible for creating a test suite and then testing the software to check that it passes all the tests. You have probably noticed that practice websites, like HackerRank, implement a suite of test cases. A table is provided to help you get started with creating a test suite for this application. Remember: constructing such a test suite *is the responsibility of the software engineer* as well as writing the code!

(5) Fill in the table below with cases that show you have **completely tested** your program. A table of inputs to test is referred to as a **test suite**. What *other* possibilities should you test?

A	B	C	$root_1$	$root_2$	Mention situation or ERROR message
1	**0**	**-4**			
1	**-2**	**1**			
1	**1**	**1**			
0	**1**	**1**			
0	**0**	**5**			

What is the next to last case (row) an equation of? _____
Does your software handle this case correctly?

What is the last case (row) an equation of? _____
Does your software handle this case correctly?

You must use a liberal use of if-else statements to correctly handle all the situations.

Your program must have an initial "comment box section", giving your name, summary, and a record of your work (that is, make edits in Data Last Modified) as shown in class and examples. For your summary, the opening, first sentence *must* be a clear description of what this app does, *not how it does it* (that is, don't write about if-else statements). A second section must include the INPUT that the program uses (e.g. three Real coefficients from `stdin`) and a description of the program's OUTPUT (indicate *where* the output appears, in this case, `stdout`).

- You *must* comment *each* variable. Variables *must* have good names.
- You *must* use constants where appropriate. Constants should be named using all UPPERCASE letters.
- Along with your folder of code, make a (text) file called: `README.txt`. This file should explain the assignment (briefly, perhaps use your first sentence of your Summary) and list each filename included along with a note explaining what this file contains, e.g. root.cpp – source code for root finding app. At the bottom of the README, please tell me the status of your program, for example: "All code tested and working fine.", or "Almost works, …".

- When your code is fully tested (is it *really tested?)*, make a .zip of your folder and submit *only* that .zip file via the onCourse site. Remember: the program is due on **Thursday, Sept. 9** (so you really can submit it up to early **Friday** morning at 4am, but no later: onCourse will turn off for this submission after 4am). *Always* submit something, even if it doesn't fully work.