1 Submission Instructions

The project must be submitted through BBLearn as a Jupyter notebook. Create a new notebook in Google Colab. Please name your notebook as follows:

Project 1 - Your Name - Your CSU ID

For example, Alice Mathperson with ID 1234567 would name her notebook:

Project 1 - Alice Mathperson - 1234567

You are allowed to make multiple submissions in BBLearn. However, only the final submission will be graded. In your final submission, make sure of the following:

- The notebook must run without errors. Before submitting the notebook, check that all cells execute correctly, *in the order they appear in the notebook*.
- There is one exception to the rule above. Sometimes it is useful that we know, or suspect, to produce errors. In this case, provide an explanation in a text cell for the error.
- Add text cells to explain what you are doing in each step, and state your conclusions. Your project should be readable as a mathematical paper.
- Your code should not only be correct, but well structured and clear. Use meaningful variable names and add comments where you feel necessary.

2 Project Statement

Let's suppose that we need to repeatedly solve quadratic equations, for a large number of cases. In this project, you will define a function `quadratic_equation` that outputs the solutions to a the quadratic equation $ax^2+bx+c=0$ given the coefficients a, b c. We start with a reasonably simple implementation, and then add features to handle more complicated cases. At the end, we will have professional quality code to solve quadratic equations.

2.1 Part 1 — Define and test your function

Write a function that implements the quadratic formula:

$$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Your code will have the following structure:

```
def quadratic_equation(a, b, c):
    ... formula breakdown ...
    x1 = ...
    x2 = ...
    return x1, x2
```

It is recommended that long formulas are avoided, by doing the computation in a series of statements. For example, the function can start by computing the discriminant $b^2 - 4ac$, and then using this partial computation in the evaluation of the solutions.

Test your code with at least ten test cases. Each test case consists of values for the coefficients a, b, c. Make sure to include tests for the following cases:

- The solutions are real and distinct.
- The solutions are not real.
- The two solutions are identical.
- The coefficients are complex numbers.
- The coefficient a is zero.

Make notes of any errors that occur.

2.2 Part 2 — Handling Complex Solutions

We now want to polish our function in a way that it produces correct and meaningful results in all cases. We want a function that works in any case, and returns the right type of solution, specially regarding the issue of complex coefficients and/or solutions.

Use the following structure for the function definition:

```
def quadratic_equation(a, b, c):
    if type(a) == complex or type(b) == complex or type(c) == complex:
        ... computation in the case of complex coefficients ...
        x1 = ...
        x2 = ...
    else:
        delta = b ** 2 - 4 * a * c
        if delta >= 0:
            ... real coefficients, but complex roots ...
            x1 = ...
            x2 = ...
        else:
            ... real coefficients, real roots ...
            x1 = ...
            x2 = ...
return x1, x2
```

Test the new version of the function with the same test cases you used in Part 1.

2.3 Part 3 — Handling Exceptions

At this point, there is still one case that your function does not handle well: if the coefficient a is zero. To take care of this case, we will make the function signal an error with a meaningful message. Add the following code right after the `def quadratic_equation(a, b,c):`

```
if a == 0:
    raise ValueError("coefficient a cannot be zero")
```

Repeat all tests with the new version of the function.