**Technical Report SY and Rivera**

**Overview**

The project aims to build a Java program that solves quadratic expressions in all cases, including when the roots are complex numbers. The program includes classes for Comp, CompPair, Quad, and SolveEq, along with additional methods to determine whether a complex object is real and whether both complex numbers are identical. The main method or SolveEq is built to solve several examples of quadratic equations input from the keyboard and output comments on the types of roots obtained.

**Approach**

We started by analyzing the project requirements and discussing the best approach to addressing them. Given the complexity of the task, we decided that the Quad class would be the most effective place to start. This allow us to build and test our secondary calculator before proceeding to the next class. We have implemented the methods and variables required to solve quadratic equations in the Quad class, including computing the discriminant to determine the properties of the roots. Then we tested the calculator to make sure it was working properly. With the Quad class working, we shifted our focus to the Comp and CompPair classes. We implemented the necessary variables and methods in both classes, such as determining if a complex number object is real and if both complex numbers are identical. Finally, We created a SolveEq class with a main method that prompts the user for input, creates a Quad object, and calls the solve() method to get the root. Then we checked the type of route and printed the result.

Taking a top-down approach to the project allowed us to track progress and make sure each class was working properly before moving on to the next. With GitHub, we were able to collaborate effectively and each team member was always aware of the other's progress. We only had one meeting to finalize the code and make sure it was working correctly before writing this report.

**Program Itself**

The program consists of four classes, each with its specific methods, variables, and data structures.

* The Comp class contains a method to determine whether a complex object is real: It contains two double variables, real and fake, representing the real and imaginary parts of the complex number. The class also has two methods, new\_Format() and neg\_Format(), used to print the complex roots in different formats. The constructor initializes the real and fake values of the complex number.
* Top of Form
* Bottom of Form
* CompPair class contains a method to determine whether both roots are complex: The checker() method takes in two double variables representing the real and imaginary parts of a complex number and returns a string indicating if both roots are the same or different. The real\_root() method also takes in the same variables and returns the real part of the root if the roots are identical and 0 representing a false case.
* The Quad class contains a method to solve the quadratic equation and return a CompPair object: We used different variables (double x: to output the result. a, b and c: store the coefficient of x^2, x, and the constant term respectively). Also, it extends the CompPair class and includes methods to compute the discriminant and solve for the roots. The compare() method uses the CompPair class to check if the roots are identical.
* The SolveEq program uses the Quad class to solve quadratic equations by computing the discriminant and using the quadratic formula to find the roots. It also checks for the type of equation (linear or quadratic) and formats the coefficients into a standard formula. The program outputs the discriminant, standard formula, and roots of the equation.

**Lessons learned**

From what we learned in class, we used a lot of methods, super keywords, data type, Boolean statements, if else statements (main method), public vs private vs static, class attributes, class inheritance. The program helped us to understand the importance of testing and debugging code. Overall, we have enjoyed as it helped us strengthen our knowledge in object oriented programming.