Link to the repository
https://github.com/Scoop-A-Doop/cs2300-coursework

Pseudocode
Part1:

- Declare final variables for file writing and decimal format converters.
- Main: Establish all needed variables and call all functions.
- PopulateMatrixes: Extract the direction of point, point on plane and normal vector information from the input file (everything in line 1).
- readFile: Reads all of the lines of the input file (except line 1, populateMatrixes does that), extracting/saving all of the information.
- organizePoints: Takes the information obtained from readFile and organizes them into matrices.
- parallelProjection: Carries out all of the calculation to find the parallel projection. (reference off of the class lecture for this).
- perspectiveProjection: Carries out all of the calculations to find the perspective projection. (reference off of the class lecture for this).
- writeToFile: Writes the information from parallel projection and perspective projection into different files "sshouib_part1_output_A.txt" and "sshouib_part1_output_B.txt". A is parallel projection (subpart 1), B is perspective projection (subpart 2).

Part2:

- Declare final variables for file writing and decimal format converters.
- Main: Establish all needed variables and call all functions.
- subPartOneReadFile: Reads the input file to obtain all of the information needed to go through the calculations for sub part one, and also calculate the equation of the plane.
- distanceCalculator: Using the data from subPartOneReadFile, calculate the distance using the distance formula.
- subPartTwoReadFile: Reads the input file to obtain all of the information needed to go through the calculations for sub part two, and also calculate some information such as the plane equation.
- intersectionCalculator: Using the data from subPartTwoReadFile, calculate the intersection point.
- writeToFile: Writes the information from parallel projection and perspective projection into different files "sshouib_part2_output_A.txt" and "sshouib_part2_output_B.txt". A is distance calculation (subpart 1), B is intersection calculation (subpart 2).

Part3:

- Note: The calculation procedure is accurate, however due to the nature of programming languages, the results might be slightly off. However, the process itself is 100% accurate to how it should be.
- Declare final variables for file writing.
- Main: Establish all needed variables and call all functions.
- calculateEigenVectors: Calculates the dominant eigenvector using the power method (reference off of the class lecture for this). For the sake of consistency and accuracy, I

have not implemented any decimal formatting at any point in the calculating process. This can result in slight variations in answers, however the calculating process itself is completely accurate.

- sortWebPageRank: Create an array sorted in numerical order (1 - N). This will mimic the current standing webpage indices of the eigenVector matrix. Then, sort the eigenvector calculated from calculateEigenVectors while also moving the other array we created that contains the webpage indices in the same way the eigenVectors are being sorted, giving you the ordered rank of the webpage from highest to lowest.
- obtainMatrixBoundaries: Figures out what the boundaries are for the N*N google matrix. Because the file is going to be NxN, you only need to know the number of rows, since rows = columns, so only read the row.
- readFile: Populate the google matrix given the input file.
- isValidMatrix: Makes sure the matrix is indeed full, (so there are N*N elements in the matrix), makes sure there are no negative numbers, and makes sure the matrix is stochastic. Note: Adding in Java is sometimes NOT accurate. One row for any given row may calculate to 1.0 in any normal calculator, but in java, it might result in 1.000002. Because of this, I add a +-0.1 leniency, so as long as the columns per row calculate from 0.99 < x < 1.1, then it will consider the row to be stochastic. This is the only way I am able to do this in Java, as there are limited options when it comes to rounding. This may result in slight variations in answers moving forward, however the calculating process itself is completely accurate.
- writeToFile: Writes the information from parallel projection and perspective projection into different files "sshouib_part3_output.txt", line 1 being the eigenvector, line 2 being the ranked webpage indices from highest to lowest ran.