# Assignment # 4

1. The ForwardElimination algorithm fails for the given system because it does not use pivoting, which is crucial for ensuring numerical stability when solving systems of linear equations. Pivoting involves swapping rows to place the largest absolute value in the pivot position, preventing issues such as division by zero or amplification of rounding errors. Without this, the algorithm might select a poor pivot element, such as a zero or very small number, leading to incorrect row operations and altering the system in a way that distorts the original solution. In this case, the algorithm fails to properly reduce the equations properly, even though the correct solution, (x1,x2,x3)=(1,2,3)(x_1, x_2, x_3) = (1, 2, 3)(x1,x2,x3)=(1,2,3), can easily be verified. The BetterForwardElimination algorithm addresses this by incorporating partial pivoting. This strategy ensures that the equations are transformed accurately while maintaining their dependencies, avoiding numerical instability, and reliably reducing the system to a solvable form. As a result, it can correctly determine the solution without errors.

2. The BetterForwardElimination algorithm fails to solve this system because it doesn't handle cases where some of the equations are **linearly dependent**, even though the solution x1=1,x2=2,x3=3x_1 = 1, x_2 = 2, x_3 = 3x1=1,x2=2,x3=3 can easily be checked. The system is:

$$
\begin{aligned}
x1 + x2 + x3 &= 6 \\
x1 + x2 + 2x3 &= 9 \\
x1 + 2x2 + 3x3 &= 15
\end{aligned}
$$

In this case, the third equation is actually a combination of the first two—it's equivalent to 2×(row 1)+(row 2). During forward elimination, this dependency causes a row of zeros to appear in the augmented matrix. The algorithm sees this zero row and assumes there isn't enough information to find a unique solution, even though the solution exists. The algorithm is designed to handle issues with pivoting and numerical stability, but it doesn't have a way to recognize or deal with rows that are dependent.

To fix this problem, the algorithm should include a way to detect and handle linear dependencies. For example, it could check if any rows are linear combinations of others before starting the elimination process, using methods like rank calculation. If dependent rows are found, they can be removed, leaving only the independent equations to solve. After solving the reduced system, the solution should be checked by plugging it back into the original equations to make sure it works for all of them, including the dependent ones. Adding these steps would allow the algorithm to handle cases like this one and still find the correct solution.