Alessandra Giovenco
HW 4 Questions 1 and 2
CS 2223 B23

1. (5 Points) Explain why the (repaired^1) *ForwardElimination* algorithm on page 210 of Levitin fails to provide a solution for:

x1 + x2 + x3 = 6
x1 + x2 + 2x3 = 9
x1 + 2x2 + 3x3 = 14

despite the fact that x = (1, 2, 3) or x_1 = 1, x_2 = 2, x_3 = 3 can be easily verified as a solution to the system.

How does the *BetterForwardElimination* algorithm on page 211 of Levitin remedy this?

ForwardElimination might fail if the pivot elements chosen are not the largest in their columns, therefore leading to a divide by zero situation or a very small number which could cause inaccurate results (like round off errors). The *BetterForwardElimination* algorithm gets rid of this risk by choosing the largest pivot element in the column, which makes the algorithm more accurate and provides the solution where *ForwardElimination* fails.

2. (10 Points) Explain in some detail why the *BetterForwardElimination* algorithm on page 211 of Levitin fails to provide a solution for:

x1 + x2 + x3 = 6
x1 + x2 + 2x3 = 9
2x1 + 2x2 + 3x3 = 15

despite the fact that x = (1, 2, 3) or x_1 = 1, x_2 = 2, x_3 = 3 can be easily verified as a solution to the system.

What can be done to remedy this shortcoming in the algorithm?

A few potential reasons why the *BetterForwardElimination* algorithm fails to provide a solution. Though there is partial pivoting in this algorithm, there can be a loss of significant digits in floating-point arithmetic because of subtraction of almost equal numbers. You can apply scaling to the rows of the matrix before performing elimination to ensure that each row has comparable coefficients.