1.

The ForwardElimination failed to provide a solution because of

A[j,k] = A[j,k] – A[j,k] \* A[j,i] / **A[i,i]**

* When A[i][i] = 0 the formula won’t work because we cannot divide by 0 🡪 Cannot use the row as the pivot if the first element is 0 during the process of solving the matrix. In this case, when i=2, A[2,2] = 0;

The BetterForwardElimination with partial pivoting remedy this by finding the valid pivot row by running thru the matrix everytime I change and find the largest value of a specific position then swap the whole role to the pivot row if it is greater

for j ← i + 1 to n do if |A[j, i]| > |A[pivotrow, i]| pivotrow ← j

//find valid pivot row by taking the greatest value

for k ← i to n + 1 do swap(A[i, k], A[pivotrow, k])

//then swap to the right position of the pivot row

2.

- Although the BetterForwardElimination avoid the possibility of dividing 0 by picking a valid pivot row, it is failed at the same mistake when the system have free variable 🡪 not only the lower triangle = 0 but also the other position can also be 0.

- In this situation x1 and x2 are free variable and while finding pivot point for x2, since the 2 out of 3 equation became identical, 🡪 0 became pivot point once again which make the equation fail again

🡪 BetterForwardElimination work for unique solutions but not when the system has free variable

* To avoid this, we can check for special case (free variable or infinite answer) before moving on the find the formula
* Can be done by checking for A[i][i] and make sure it is not 0 before divided A[j][i] to it 🡪 have another function to deal with it or just send out messages