Problem 7.1:

Problem 7.2:

Problem 5.3 [Modified]:

- Do part (a).
- Write a Matlab function `LUNoPivot` that takes as input a square matrix and returns two matrices L and U, lower and upper triangular matrices such that L has 1's on the diagonal and such that $A = LU$. Do not pivot (i.e., do not perform row interchanges). You can use the code on page 140 of your text as a starting point. You should test your code on the 3×3 matrix presented in class today; the matrix A from page 135. That is, verify that indeed $LU = A$.

  Note that the code on page 140 is being sneaky. Rather than building two matrices, it builds just one. Since L always has 1s on the diagonal, it only has interesting entries below the diagonal. And since U is all zeros below the diagonal, there's space there to store the entries of L! This is an important space saving technique when the matrices involved are large: no need to go around working with extra matrices that are half zeros and use up twice the needed storage. But for the purposes of this exercise and clarity, we'll return L and U seperately.
- Now do part (c). You'll need to use `lsolve` from the text (page 140) and `usolve` from Problem 7.2.

Problem 7.4: The matrix P in this problem is called a permutation matrix. We'll discuss this more when we cover pivoting. But you can still work on this problem. The first step to solving $Ax = b$ in this context is to multiply the equation by P. Notice that all P does in rearrange the entries of b: that's why it's called a permutation matrix!

Problem 7.6: