Problem 7.3 [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.

Supplemental 1: Write a function to compute the inverse of a $n \times n$ matrix A as follows.

(a) Let $\mathbf{b}_i$ be column i of A^{-1}. What are the entries of $A\mathbf{b}_i$? Hint: most of them are zero! Use the column perspective of matrix multiplication.

(b) Call your `LUNoPivot` code (or better code with pivoting!) to get L and U (and P if you want!).

(c) For each i, compute column $\mathbf{b}_i$ of A^{-1} using the strategy of part a). For each column, you will call your `lsolve` and your `usolve` funtions exactly once.

Supplemental 2: Determine, with justification, the number of floating point operations required to compute the inverse of a matrix using the strategy of the previous problem. A complete answer will be of the form

$$cn^j + O(n^k)$$

where c is an explicit number, and where j and k are explicit integers with $j > k$.

Supplemental 3: How many 6×6 permutation matrices are there? A complete answer will justify the number.

Supplemental 4: A permutation matrix can be represented by a vector $[p_1, \ldots, p_n]$ where p_i records which column contains the 1 in row i.

Modify the code for `lsolve` to make a new function `plsolve` so that it takes as arguments

1. `P`, an n-dimensional vector representing a permuation matrix,
2. `L`, a lower triangular $n \times n$ matrix with 1's on the diagonal.
3. `b` an n-dimensional vector

It should return the solution to $L\mathbf{c} = P\mathbf{b}$.

Test your code on problem 15 of the `WSPartialPivoting` worksheet. That is, you will type in the matrics U, L you determined on the worksheet along with a vector P representing the permutation matrix. Then use your brand new `plsolve` along with your older `usolve` to compute the solution of $A\mathbf{x} = \mathbf{b}$. You should verify that the $\mathbf{x}$ that you compute really works by multiplying by A!