Problem 7.1: Write the following matrix in the form LU, where L is a unit lower triangular matrix and U is an upper triangular matrix,

$$\begin{pmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix}$$

Step one of gaussian elimination by reducing the terms of the first column,

$$L_1A = \begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{4} & 1 & 0 \\ \frac{1}{4} & 0 & 1 \end{pmatrix}\begin{pmatrix} 4 & -1 & -1 \\ -1 & 4 & -1 \\ -1 & -1 & 4 \end{pmatrix} = \begin{pmatrix} 4 & -1 & -1 \\ 0 & \frac{15}{4} & -\frac{5}{4} \\ 0 & -\frac{5}{4} & \frac{14}{4} \end{pmatrix}.$$

Step two we reduce the terms in the second column and we get L and U,

$$L_2L_1A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{3} & 1 \end{pmatrix}\begin{pmatrix} 4 & -1 & -1 \\ 0 & \frac{15}{4} & -\frac{5}{4} \\ 0 & -\frac{5}{4} & \frac{14}{4} \end{pmatrix} = \begin{pmatrix} 4 & -1 & -1 \\ 0 & \frac{15}{4} & -\frac{5}{4} \\ 0 & 0 & \frac{10}{3} \end{pmatrix}.$$

Therefore we know that,

$$L = L_1^{-1}L_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -\frac{1}{4} & 1 & 0 \\ -\frac{1}{4} & -\frac{1}{3} & 1 \end{pmatrix}$$

$$U = \begin{pmatrix} 4 & -1 & -1 \\ 0 & \frac{15}{4} & -\frac{5}{4} \\ 0 & 0 & \frac{10}{3} \end{pmatrix}$$

Now consider the following equation,

$$A = LL^T.$$

Through matrix multiplication we see that LL^T results in the following symmetric matrix.

$$LL^T = \begin{pmatrix} L_{11}^2 & & \text{(symmetric)} \\ L_{21}L_{11} & L_{21}^2 + L_{22}^2 & \\ L_{31}L_{11} & L_{31}L_{21} + L_{32}L_{22} & L_{31}^2 + L_{32}^2 + L_{33}^2 \end{pmatrix}.$$

Plugging in our values from A and solving for the L_{ij} terms we get,

$$L = \begin{pmatrix} 2 & 0 & 0 \\ -\frac{1}{2} & \frac{\sqrt{15}}{2} & 0 \\ -\frac{1}{2} & -\frac{\sqrt{15}}{6} & \frac{\sqrt{30}}{2} \end{pmatrix}$$

Therefore writing A in terms of its LL^T factorization,

$$A = \begin{pmatrix} 2 & 0 & 0 \\ -\frac{1}{2} & \frac{\sqrt{15}}{2} & 0 \\ -\frac{1}{2} & -\frac{\sqrt{15}}{6} & \frac{\sqrt{30}}{2} \end{pmatrix}\begin{pmatrix} 2 & -\frac{1}{2} & -\frac{1}{2} \\ 0 & \frac{\sqrt{15}}{2} & -\frac{\sqrt{15}}{6} \\ 0 & 0 & \frac{\sqrt{30}}{2} \end{pmatrix}$$

We can also double check our algebra using Matlab,
Console:

```
>> L = [2 0 0; -(1/2) ((15)^(1/2))/2 0; (-1/2) -((15)^(1/2))/6
((30)^(1/2))/3]

L =

   2.000000000000000                   0                   0
  -0.500000000000000   1.936491673103709                   0
  -0.500000000000000  -0.645497224367903   1.825741858350554

>> L*L'

ans =

     4    -1    -1
    -1     4    -1
    -1    -1     4
```

Problem 7.2: Write a function *usolve*, analogous to the function *lsolve* in section 7.2.2. to solve an upper triangular system, $Ux = y$.

Code:

```
function X = usolve(U, b)
 %This funciton takes in an upper triangular matrix,
 %and an appropriate resultant vector and returns a
 %solution, by back substitution.

n = size(b,2); % Pulling dimensions of U and b

X = [];% initilizing soution vector


X(1) = b(n)/ U(n,n); % Solving for nth term in the solution vector


% This nested for loop iterates through
%the upper triangular matrix. Pulls
% the values from the matric not on
% the diagonal and calculated the
% solution vector.
```

```
for i = n-1:-1:1 % n-1 to 1 rows to iterate through
    a = []; % Initilizing vector to pull terms from the ith row of U
    for j = i+1:n % Iterating through ith row of U and pulling non zero te
        a = [a,U(i,j)];
    end

X = [(b(i)-(dot(X,a)))/U(i,i), X]; %Calculating X(i)
end
```

Testing the code with a couple of problems,

Console:

```
>> U = [1  3  8  3  1;0  7  5  3  1;  0  0  1  3  4;  0  0  0  12  3;  0  0  0  0  3]

U =

     1     3     8     3     1
     0     7     5     3     1
     0     0     1     3     4
     0     0     0    12     3
     0     0     0     0     3

>> b = [2  2  2  1  2]

b =

     2     2     2     1     2

>> X = usolve(U, b)

X =

    3.3452    0.5238   -0.4167   -0.0833    0.6667

>> Y = linsolve(U,b')

Y =

    3.3452
    0.5238
   -0.4167
```

```
-0.0833
 0.6667
```

Problem 7.3 [Modified]:

- By hand, solve the following linear system exactly,

$$A = \begin{pmatrix} 10^{-16} & 1 \\ 1 & 1 \end{pmatrix}, b = \begin{pmatrix} 2 \\ 3 \end{pmatrix}.$$

  Write your answer in the form that it is clear what the approximate values of x_1 and x_2 are.

  Solution:
  First we will perform gaussian elimination on the augmented matrix Ab.

$$\begin{pmatrix} 10^{-16} & 1 & |2 \\ 1 & 1 & |3 \end{pmatrix} \rightarrow \begin{pmatrix} 10^{-16} & 1 & |2 \\ 0 & 1 - 10^{-16} & |3 - 2 * 10^{-16} \end{pmatrix}$$

  Solving the system of linear equation, we get,

$$x_2 = \frac{3 - 2 \times 10^{-16}}{1 - \times 10^{-16}} \approx \frac{-2 \times 10^{-16}}{- \times 10^{-16}} \approx 2$$

$$x_1 = (2 - \frac{3 - 2 \times 10^{-16}}{1 - \times 10^{-16}})10^{16} \approx (2 - \frac{-2 \times 10^{-16}}{- \times 10^{-16}})10^{16} \approx (2 - 2)10^{16} \approx 0$$

- 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 separately.

Code:

```
function [L,A] = LUNoPivot(A)
%This funciton takes an NxN matrix A and returns an LU factorization
% without pivoting the rows
n = size(A,2);
L = zeros(n);


for k = 1:n %Initializes the diagonal of L
    L(k,k) = 1;
end


for i = 1:n-1 % Iterates through columns of A

    if A(i,i) == 0
     error('Cannot LU factorize without pivoting')
    end

    for j = i+1:n % Iterates through Rows of A

        x = A(j,i)/A(i,i); %Calculates factor for Gauss Elim

        L(j,i) = x; %Stores factor in L

        for k = 1:n % Iterates through current row and performs Gaus
            A(j,k) = A(j,k) - (A(i,k)*x);
        end

    end

end
```

Testing our code with the 3x3 matrix presented in class.

Console:

```
>> A = [1 2 3; 4 5 6; 7 8 0]

A =
```

```
        1     2     3
        4     5     6
        7     8     0

>> [L,U] = LUNoPivot(A)

L =

        1     0     0
        4     1     0
        7     2     1


U =

        1     2     3
        0    -3    -6
        0     0    -9

>> L*U

ans =

        1     2     3
        4     5     6
        7     8     0
```

- Use `lsolve` from the text (page 140) and write matlab that solves the linear system $Ax = b$. Compare the answer to this code to the one you determined by hand.

  Solution:
  We simply write a function that calls our prior three functions appropriately to produce a solution. Consider,

  Code:

```
function x = NoPivotSolve(A, b)
%This funciton solves a linear system where A
%is NxN and LU factorization requires no pivoting

[L,U] = LUNoPivot(A);
```

```
y = lsolve(L,b);

x = usolve(U,y);
```

Using our code to solve $Ax = b$

Console:

```
>> A = [10e-16 1; 1 1]

A =

    0.000000000000001    1.000000000000000
    1.000000000000000    1.000000000000000

>> b = [2,3]

b =

      2      3

>> NoPivotSolve(A,b)

ans =

    1.110223024625157    1.999999999999999
```

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!

Solution:
Consider the following,

$$Ax = b$$
$$PAx = Pb.$$

Note that we have the LU factorization for the matrix PA therefore all we must do is solve for Pb and continue with solving the system by LU factorization as normal. Through matrix multiplication,

$$Pb = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2 \\ 10 \\ -12 \end{pmatrix} = \begin{pmatrix} -12 \\ 2 \\ 10 \end{pmatrix}.$$

Then we continue by solving the system $L\hat{b} = Pb$,

$$\begin{pmatrix} 1 & 0 & 0 \\ \frac{1}{2} & 1 & 0 \\ \frac{1}{3} & \frac{1}{4} & 1 \end{pmatrix} \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \end{pmatrix} = \begin{pmatrix} -12 \\ 2 \\ 10 \end{pmatrix}.$$

Doing so we get,

$$\hat{b} = \begin{pmatrix} -12 \\ 12 \\ 8 \end{pmatrix}.$$

Finally we solve the system by solving $Ux = \hat{b}$

$$\begin{pmatrix} 2 & 3 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -12 \\ 12 \\ 8 \end{pmatrix}.$$

Therefore the solution to $Ax = b$ is,

$$x = \begin{pmatrix} -14 \\ 4 \\ 4 \end{pmatrix}$$

Problem 7.6: How many operations are requresd to computre the following,

1. Compute the sum of two n-vectors?

   Solution:
   When we add or subtract vectors we add or subtract the correspoding socmponents therefore there are $2n$ operations when computing the sum of two n-vectors.

2. Compute the product of an m by n matrix with an n-vector?

Solution:
To compute the product of a matrix and a vector, we take the dot product of each row vector from the matrix with the vector. Note that the dot product of two n-vectors has n multiplications and $n-1$ additions, and in this case there are m dot products. Thus the total number of computations is, $m(2n-1)$.

3. Solve an n by n upper triangular linear system $Ux = y$?

Solution:
Recall that we did this exercise in class. Consider the form of x_1,

$$x_1 = \frac{y_1 - a_2x_2 - a_3x_3 - a_4x_4... - a_nx_n}{a_1}$$

where $a_i = U(1,i)$. Now note that for each x_n computation there are $n-1$ multiplications, $n-1$ subtractions, and 1 division so $2n-1$ operations total. To solve the whole system we must compute all x_n therefore the total number of computations is,

$$\sum_{i=1}^{n} 2i - 1 = n^2.$$

Note that the algebra for this computation was done in class and on the worksheet.