The written document focuses on explaining how my program is executed to solve $A\mathbf{x} = \mathbf{b}$ using Gaussian Elimination on page 67 of the textbook.

Since we are given a fact that the matrix A can be factorized to a lower-triangular and a upper-triangular matrix L and U, which is $A = LU$. The first step is to decompose A.

Step 1: Decompose A to L and U

(1) Using gaussian Elimination by obtaining elementary matrices(elimMat.m)

```
function [M_k,L_k] = elimMat(A, k)

[n,~] = size(A);

M_k = eye(n);
% Initialize M_k as an identity matrix

M_k_entry = -A(k+1:n, k)/A(k,k);

%Vectorization

M_k(k+1:n, k) = M_k_entry;
%Obtain the entries in the lower triangular of M_k

L_k = inv(M_k);
%Obtain L_k according to the formula that L_k = inv(M_k)

end
```

Every time when computing an elementary matrix, multiply it to matrix A on the left hand side until we derive a upper-triangular matrix by doing so. We know that

$$A = LU = LM_{n-1}M_{n-2} \cdots M_1 A$$

Thus, we can derive that $L = (M_{n-1}M_{n-2} \cdots M_1)^{-1} = {M_1}^{-1}{M_2}^{-1} \cdots {M_n}^{-1}$

(2) Therefore, we can simultaneously update the temporary matrix L by inversing the elementary matrix we have obtained.

```
function [L, U] = myLU(A)

[n,~] = size(A);
L = eye(n);
U = A;

for k = 1:n-1
    %The reason for the termination of for loop is (n-1) is that
    %we start every time from (k+1)th column to get M_k entry, thus k+1 <=n
    %
    [M_k, L_k] = elimMat(U, k);
    U = M_k * U;
    L = L* L_k;
end
end
```

After LU decomposition, we get a new logically equivalent equation $LU\mathbf{x} = \mathbf{b}$.

Step 2: Let $\mathbf{y} = U\mathbf{x} \Rightarrow L\mathbf{y} = \mathbf{b}$

Forward Substitution to solve $\mathbf{y}$

Since zeros occupy the top right portion of L, we can solve y_1 first from the top, which is forward substitution.

```
function y = fwdSubst(L, b, k)
%Foward substitution
[m,n]=size(L);
if ~exist('k')  % If first call no k param given, but k=1
    k=1;
end

y=b(k)/L(k,k);
if k < n % Recursion step
    l = [zeros(k,1);L(k+1:m,k)];
    y = [y;fwdSubst(L,b-y*l,k+1)];
end
```

Step 3: $U\mathbf{x} = \mathbf{y}$ Backward Substitution to solve $\mathbf{x}$

We have $U\mathbf{x} = \mathbf{y}$ and $\mathbf{y}$ has been derived in the previous step. Likewise, since zeros occupy the bottom left portion of U, we can solve x_n first from the top, which is forward substitution. We can obtain the equation system below:

$$
\begin{gathered}
u_{11}x_1 + u_{12}x_2 + \cdots + u_{1n}x_n = y_1 \\
u_{22}x_2 + \cdots + u_{2n}x_n = y_2 \\
\vdots \\
u_{nn}x_n = y_n
\end{gathered}
$$

We can derive that $x_n = \frac{y_n}{u_{nn}}$, iterating x_n to equations above, we can obtain that

$x_{n-1} = \frac{y_{n-1} - u_{(n-1)n}x_n}{u_{(n-1)(n-1)}}$, $x_{n-2} = \frac{y_{n-2} - (u_{(n-2)(n-1)}x_{n-1} + u_{(n-2)n}x_n)}{u_{(n-2)(n-2)}}$

Thus, we can induct the general formula for x_i, which is

$$x_i = \frac{y_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}$$

```
function x = backSubst(U, y, k)

x = zeros(k, 1);

for i=k:-1:1
    x(i)=(y(i)-U(i, i+1:k)*x(i+1:k)) / U(i, i);
end
end
```

Finally, we can solve $\mathbf{x}$ after executing the program.