3. Write a program for solving an $n \times n$ system of equations by the naive Gauss elimination (see the pseudocode on p.77). (a) Define an $n \times n$ array by $a_{ij} = -1 + 2\min\{i, j\}$. Then setup the array $(b_i)$ in such a way that the solution of the system $\sum_{j=1}^{n} a_{ij}x_j = b_i, 1 \leq i \leq n$ is $x_j = 1, 1 \leq j \leq n$. Test your program on this system for a few values of $n$, say $n = 10, 20, 100$, and comment on the accuracy obtained. In particular, evaluate the error $\|x - \tilde{x}\|_\infty$, where $\tilde{x}$ is the obtained solution, and where, as usual, $\|y\|_\infty := \max\{|y_1|, \ldots, |y_n|\}$ for $y = (y_1, \ldots, y_n)$. (b) Repeat the same experiment as in part (a), but with matrix with entries $a_{ij} = \frac{1}{i+j-1}, i, j = 1, \ldots, n$ (this matrix is called a Hilbert matrix). Discuss your results.

(a) Creating the program letting it solve the system of equations we get:

```
>> P3_Gaussian_Elimination
For n = 10, we have an error = 0.000000000000000
For n = 20, we have an error = 0.000000000000000
For n = 100, we have an error = 0.000000000000000
```

So Naïve Gaussian elimination successfully solves this system even for large values, as we have no small pivot elements that lead to large multipliers and large roundoff errors.

(b) Running the Naïve Gauss program on the Hilbert Matrix we have:

```
>> P3_Gaussian_Elimination
For n = 10, we have an error = 0.000436911529728
For n = 20, we have an error = 58.516277170010319
For n = 100, we have an error = 224.706251805224355
```

The errors are significantly larger, probably because the matrix is ill-conditioned (large condition number)

Code:

```matlab
for n = [10, 20, 100]
    for i = 1:n
        for j = 1:n
           % A(i,j) = -1 + 2*min([i j]);
            A(i,j) = 1/ (i+j-1);
        end
    end
    x_exact = ones(n, 1);
    b = A*x_exact;


    % (2) Naive Gaussian Elimination

    % (2.1) Forward elimination
    for k = 1:(n-1)
        for i = k+1:n
            xmult = A(i,k) / A(k,k);
            %A(i,k) = xmult;

            for j = k:n
                A(i,j) = A(i,j) - xmult*A(k,j);
            end
            b(i) = b(i) - xmult*b(k);
        end
    end
    x(n,1) = b(n) / A(n,n);

    % (2.2) Back substitution
    for i = n-1:-1:1
        sum = b(i);
        for j = i + 1:n
            sum = sum - A(i,j)*x(j);
        end
        x(i) = sum / A(i,i);
    end


    % (2.3) Evaluating error
    error = x_exact - x;
    error_inf = max(abs(error));
    fprintf("For n = %d, we have an error = %.15f\n", n, error_inf)
end
```