In [2]:
```python
# Import the NumPy library
import numpy as np

# Define the function that performs Gaussian elimination with partial pivoting
def gaussian_elimination_with_pivoting(A, b):
    n = A.shape[0]
    U = np.copy(A)
    L = np.eye(n)
    P = np.eye(n)

    # Forward Elimination with Partial Pivoting
    for k in range(n):
        # Find the row with the maximum absolute value for the pivot column
        max_row = np.argmax(np.abs(U[k:n, k])) + k
        # Swap the current row and the max row for U, P, and L
        U[[k, max_row]] = U[[max_row, k]]
        P[[k, max_row]] = P[[max_row, k]]
        L[[k, max_row], :k] = L[[max_row, k], :k]

        # Check if the matrix is singular
        if U[k, k] == 0:
            return "Matrix is singular", None, None, None

        # Eliminate elements below the pivot to create zeros
        for i in range(k+1, n):
            factor = U[i, k] / U[k, k]
            L[i, k] = factor
            U[i, k:] = U[i, k:] - factor * U[k, k:]

    # Forward substitution to solve Ly = Pb
    y = np.zeros(n)
    b = np.dot(P, b)
    for i in range(n):
        if i == 0:
            y[i] = b[i, 0]  # Directly assign the first element
    else:
            y[i] = b[i, 0] - np.dot(L[i, :i], y[:i])

    # Backward substitution to solve Ux = y
    x = np.zeros(n)
    for i in range(n-1, -1, -1):
        x[i] = (y[i] - np.dot(U[i, i+1:], x[i+1:])) / U[i, i]

    # Return the solution x, and the matrices P, L, and U
    return x.reshape(-1, 1), P, L, U
```

In [3]:
```python
# Define the matrix A and B from the statement.
A_b = np.array([
    [5, 1, 0, 2, 1],
    [0, 4, 0, 1, 2],
    [1, 1, 4, 1, 1],
    [0, 1, 2, 6, 0],
    [0, 0, 1, 2, 4]
])

b_b = np.array([
    [1],
    [2],
    [3],
    [4],
    [5]
])

# Use the Gaussian Elimination function
x_b, P_b, L_b, U_b = gaussian_elimination_with_pivoting(A_b, b_b)

# show the result
x_b, P_b, L_b, U_b
```

Out[3]:
```
(array([[ 0.075],
        [-0.625],
        [ 0.   ],
        [ 0.   ],
        [ 1.25 ]]),
 array([[1., 0., 0., 0., 0.],
        [0., 1., 0., 0., 0.],
        [0., 0., 1., 0., 0.],
        [0., 0., 0., 1., 0.],
        [0., 0., 0., 0., 1.]]),
 array([[1.  , 0.  , 0.  , 0.  , 0.  ],
        [0.  , 1.  , 0.  , 0.  , 0.  ],
        [0.2 , 0.  , 1.  , 0.  , 0.  ],
        [0.  , 0.25, 0.5 , 1.  , 0.  ],
        [0.  , 0.  , 0.25, 0.4 , 1.  ]]),
 array([[5, 1, 0, 2, 1],
        [0, 4, 0, 1, 2],
        [0, 0, 4, 0, 0],
        [0, 0, 0, 5, 0],
        [0, 0, 0, 0, 4]]))
```

In [4]:
```python
# Define the matrix A and B from the statement.
A_c = np.array([
    [5, 1, 0, 2],
    [0, 4, 0, 8],
    [1, 1, 4, 2],
    [0, 1, 2, 2]
])

b_c = np.array([
    [1],
    [2],
    [3],
    [4]
])

# Use the Gaussian Elimination function
x_c, P_c, L_c, U_c = gaussian_elimination_with_pivoting(A_c, b_c)

# show the result
x_c, P_c, L_c, U_c
```

Out[4]:
```
('Matrix is singular', None, None, None)
```