In [1]:

```
%run CPL_Library.ipynb  # Running the entire library in one line because
                        # importing does not work with JupyterLab
```

Question 1

In [2]:

```
# LU decomposition using Doolittle's condition L[i][i]=1

print("The matrix is: ")
A1,ro,co = read_matrix('As4matrixA.txt')
print_matrix(A1,ro,co)

vector=[6,-3,-2,0]

# partial pivoting to avoid division by zero at pivot place
A1, vector = partial_pivot_LU(A1, vector, ro)
A1 = LU_doolittle(A1,ro)
print("The transformed LU matrix is ")
print_matrix(A1,ro,ro)

x = [0 for i in range(ro)]

x = for_back_subs_doolittle(A1,ro,vector)

print("Solutions are : ")
for i in range(ro):
    print("x["+str(i)+"] = "+str(x[i]))
```

```
The matrix is:

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    -1.0    0.0

2.0    1.0    3.0    -2.0


The transformed LU matrix is

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    2.0    -2.0

2.0    1.0    1.5    -3.0


Solutions are :
x[0] = 1.0
x[1] = -1.0
x[2] = 1.0
x[3] = 2.0
```

In [3]:

```
# LU decomposition using Crout's condition U[i][i]=1

print("The matrix is: ")
A2,ro,co=read_matrix('As4matrixA.txt')
print_matrix(A2,ro,co)

vector=[6,-3,-2,0]

# partial pivoting to avoid division by zero at pivot place
A1, vector = partial_pivot_LU(A1, vector, ro)
A2=LU_crout(A2,ro)
print("The transformed LU matrix is ")
print_matrix(A2,ro,ro)

x = [0 for i in range(ro)]

x=for_back_subs_crout(A2,ro,vector)

print("Solutions are : ")
for i in range(ro):
    print("x["+str(i)+"] = "+str(x[i]))
```

```
The matrix is:

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    -1.0    0.0

2.0    1.0    3.0    -2.0


The transformed LU matrix is

1.0    0.0    1.0    2.0

0.0    1.0    -2.0    0.0

1.0    2.0    2.0    -1.0

2.0    1.0    3.0    -3.0


Solutions are :
x[0] = 1.0
x[1] = -1.0
x[2] = 1.0
x[3] = 2.0
```

Question 2

In [4]:

```
print("The initial matrix is : ")
B,ro,co=read_matrix('As4matrixB.txt')
print_matrix(B,ro,ro)

C=copy.deepcopy(B) # deepcopy for unchanged matrix required for inverse

identity=get_identity(ro)

# Then partial pivoting is done for both matrix and vector.
# Then the decomposition algorithm is applied.
B, identity = partial_pivot_LU(B, identity, ro)
B=LU_doolittle(B,ro)

#print("The transformed LU matrix is ")
#print_matrix(B,ro,ro)

#Checking if inverse exists
det=determinant(B,ro)
if det == 0:
    print("Determinant = zero.\nInverse doesn't exist.")
else:
    print("The inverse is:")

    # Calculating and printing inverse
    inverse= inverse_by_lu_decomposition(C, ro)
    print_matrix(inverse,ro,ro)

    # Verification: gives indentity matrix on multiplication with original matrix
    print("Verification : ")
    mm,r,c=matrix_multiply(C,ro,ro,inverse,ro,ro)
    print_matrix(round_matrix(mm),r,c)
```

```
The initial matrix is :

0.0    2.0    8.0    6.0

0.0    0.0    1.0    2.0

0.0    1.0    0.0    1.0

3.0    7.0    1.0    0.0


The inverse is:

-0.25000000000000006    1.6666666666666672    -1.8333333333333333    0.333333
3333333333

0.08333333333333337    -0.666666666666667    0.8333333333333333    0.0

0.16666666666666666    -0.33333333333333326    -0.3333333333333333    0.0

-0.08333333333333333    0.6666666666666666    0.16666666666666666    0.0


Verification :

1.0    0    0    0

0    1.0    0    0

0.0    -0.0    1.0    0

0.0    -0.0    -0.0    1.0
```

Question 3

In [5]:

```
# Function for Cholesky decomposition

print("The matrix is: ")
C,ro,co=read_matrix('As4matrixC.txt')
print_matrix(C,ro,co)

vector=[2.20, 2.85, 2.79, 2.87]

# partial pivoting to avoid division by zero at pivot place
C, vector = partial_pivot_LU(C, vector, ro)
C=LU_cho(C,ro)
print("The transformed Cholesky matrix is ")
round_matrix(C)
print_matrix(C,ro,ro)

x=for_back_subs_cho(C,ro,vector)

print("Solutions are : ")
for i in range(ro):
    print('%.2f'%x[i])
```

```
The matrix is:

10.0    1.0    0.0    2.5    2.2

1.0    12.0    -0.3    1.1    2.85

0.0    -0.3    9.5    0.0    2.79

2.5    1.1    0.0    6.0    2.87


The transformed Cholesky matrix is

3.16    0.32    0    0.79

0.32    3.45    -0.09    0.25

0    -0.09    3.08    0.01

0.79    0.25    0.01    2.31


Solutions are :
0.10
0.20
0.30
0.40
```

In [ ]: