# Assignment 4

In [1]:
```
#All the necessary codes imported from the library

import library as lib  #importing library of functions
```

## Q1 LU decomposition using Doolittle and crout

In [6]:
```
#Doolittle algorith
M = lib.matrixtxt('input.txt')  #storing the given matrix
b = lib.matrixtxt('values.txt')  #storing the RHS of equation
print ("Using Doolittle, ")
lib.LUDoolite(M)   #calling function for LU using Doolittle
L = lib.LUDoolite.L    #storing L and U from the function
U = lib.LUDoolite.U

#calling forward backward subsitution to calculate x and printing it simultaneously
print("The values of x1,x2,x3,x4 from Doolittle algorithm are ", lib.forwardbackward(L, U, b), " respectively ")
```

```
Using Doolittle,
Lower Triangular
[1, 0, 0, 0]
[0.0, 1, 0, 0]
[0.5, 1.5, 1, 0]
[0.5, -0.5, -3.0, 1]
Upper Triangular
[2.0, 1.0, 3.0, -2.0]
[0, 1.0, -2.0, 0.0]
[0, 0, 0.5, 1.0]
[0, 0, 0, 6.0]
The values of x1,x2,x3,x4 from Doolittle algorithm are  [[1.0], [-1.0], [1.0], [2.0]]  respectively
```

In [7]:
```
#Crout's algorithm to calculate Lu
print ("\nNow using Crout, ")
M = lib.matrixtxt('input.txt')  #storing the matrix
lib.LUCrout(M)              #calling LU crout
L2 = lib.LUCrout.L           #storing L and U from the function
U2 = lib.LUCrout.U
d = lib.matrixtxt('values.txt')   #storing RHS

#calling forward backward on L and U, and simultaneously printing the result
print("The values of x1,x2,x3,x4 from Crout algorithm are ", lib.forwardbackward(L2,U2,d), " respectively ")
```

```
Now using Crout,
Lower Triangular
[2.0, 0, 0, 0]
[0.0, 1.0, 0, 0]
[1.0, 1.5, 0.5, 0]
[1.0, -0.5, -1.5, 6.0]

Upper Triangular
[1, 0.5, 1.5, -1.0]
[0, 1, -2.0, 0.0]
[0, 0, 1, 2.0]
[0, 0, 0, 1]
The values of x1,x2,x3,x4 from Crout algorithm are  [[1.0], [-1.0], [1.0], [2.0]]  respectively
```

## Q2 Calculating inverse using LU decomposition

In [4]:
```
#storing identity matrix
I = lib.matrixtxt('identity.txt')
#storing the matrix given
A = lib.matrixtxt('inverse.txt')
print ("The given matrix is " , A, "\n")

#calling the function to calculate inverse
X = lib.inverseLU(A,I)
print ("\nTo verify we multiply inverse by original matrix to get an identity matrix")
#verifying the calculated inverse by multiplying with matrix
verification = lib.multiplysquare(X,A)
```

```
The given 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]]

Lower Triangular
[1, 0, 0, 0]
[0.0, 1, 0, 0]
[0.0, 0.5, 1, 0]
[0.0, 0.0, -0.25, 1]
Upper Triangular
[3.0, 7.0, 1.0, 0.0]
[0, 2.0, 8.0, 6.0]
[0, 0, -4.0, -2.0]
[0, 0, 0, 1.5]
The inverse of the given matrix is
[0.3333333333333333, -0.25000000000000006, -1.8333333333333333, 1.6666666666666667]
[0.0, 0.08333333333333337, 0.8333333333333333, -0.6666666666666667]
[-0.0, 0.16666666666666666, -0.3333333333333333, -0.3333333333333333]
[0.0, -0.08333333333333333, 0.16666666666666666, 0.6666666666666666]

To verify we multiply inverse by original matrix to get an identity matrix
The product of
[0.3333333333333333, -0.25000000000000006, -1.8333333333333333, 1.6666666666666667]
[0.0, 0.08333333333333337, 0.8333333333333333, -0.6666666666666667]
[-0.0, 0.16666666666666666, -0.3333333333333333, -0.3333333333333333]
[0.0, -0.08333333333333333, 0.16666666666666666, 0.6666666666666666]
and
[3.0, 7.0, 1.0, 0.0]
[0.0, 2.0, 8.0, 6.0]
[0.0, 1.0, 0.0, 1.0]
```

```
[0.0, 0.0, 1.0, 2.0]
is
[1.0, 0, 0, 0]
[0, 1.0, 0, 0]
[0, 0, 1.0, 0]
[0, 0, 0, 1.0]
```

## Q3 Solving the given equation using Cholesky decomposition

In [5]:
```
B = lib.matrixtxt('cholesky.txt')   #storing the given matrix from a txt file
lib.Cholesky_Decomposition(B,4)      #calling function to use Cholesky decomposition
L = lib.Cholesky_Decomposition.L      #storing L from the function in a varialbe
U = lib.Cholesky_Decomposition.U        #storing U from the function in a varialbe
c = lib.matrixtxt('valuesC.txt')        #storing RHS of equation

#forward backward substitution of LU and printing the result
print ("The values of x1,x2,x3,x4 are: ")
for i in range (len(lib.forwardbackward(L,U,c))):
    x = lib.forwardbackward(L,U,c)[i][0]
    print (round(x,2))
```

```
Lower Triangular
[3.16, 0, 0, 0]
[0.32, 3.45, 0, 0]
[0.0, -0.09, 3.08, 0]
[0.79, 0.25, 0.01, 2.31]
Upper Triangular
[3.16, 0.32, 0.0, 0.79]
[0, 3.45, -0.09, 0.25]
[0, 0, 3.08, 0.01]
[0, 0, 0, 2.31]
The values of x1,x2,x3,x4 are:
0.1
0.2
0.3
0.4
```