Resolución de Casos 6

Problema 1

Definimos la matriz A.

```
A = [4 -1 -1 0
    -1 4 0 -1
    -1 0 4 -1
    0 -1 -1 4]
```

```
A = 4×4
     4    -1    -1     0
    -1     4     0    -1
    -1     0     4    -1
     0    -1    -1     4
```

Definimos la matriz b.

```
b = [30 40 60 70]'
```

```
b = 4×1
    30
    40
    60
    70
```

Realizamos la factorización LU.

```
[L, U] = lu(A)
```

```
L = 4×4
    1.0000         0         0         0
   -0.2500    1.0000         0         0
   -0.2500   -0.0667    1.0000         0
         0   -0.2667   -0.2857    1.0000
U = 4×4
    4.0000   -1.0000   -1.0000         0
         0    3.7500   -0.2500   -1.0000
         0         0    3.7333   -1.0667
         0         0         0    3.4286
```

Resolvemos el sistema con factorizacion LU.

```
z = linsolve(L, b)
```

```
z = 4×1
   30.0000
   47.5000
   70.6667
  102.8571
```

```
x = linsolve(U, z)
```

```
x = 4×1
   20.0000
   22.5000
   27.5000
```

```
   30.0000
```

Problema 2

Definimos nuestras variables

```
A = [4 -1 -1 0 0 0 0 0
    -1 4 0 -1 0 0 0 0
    -1 0 4 -1 -1 0 0 0
    0 -1 -1 4 0 -1 0 0
    0 0 -1 0 4 -1 -1 0
    0 0 0 -1 -1 4 0 -1
    0 0 0 0 -1 0 4 -1
    0 0 0 0 0 -1 -1 4]
```

```
A = 8×8
     4    -1    -1     0     0     0     0     0
    -1     4     0    -1     0     0     0     0
    -1     0     4    -1    -1     0     0     0
     0    -1    -1     4     0    -1     0     0
     0     0    -1     0     4    -1    -1     0
     0     0     0    -1    -1     4     0    -1
     0     0     0     0    -1     0     4    -1
     0     0     0     0     0    -1    -1     4
```

```
b = [5 15 0 10 0 10 20 30]'
```

```
b = 8×1
     5
    15
     0
    10
     0
    10
    20
    30
```

Hacemos uso del algortimo de Doolitle para factorizar A en dos matrices triangulares

```
[L, U] = doolitle(A)
```

```
L = 8×8
    1.0000         0         0         0         0         0         0         0
   -0.2500    1.0000         0         0         0         0         0         0
   -0.2500   -0.0667    1.0000         0         0         0         0         0
         0   -0.2667   -0.2857    1.0000         0         0         0         0
         0         0   -0.2679   -0.0833    1.0000         0         0         0
         0         0         0   -0.2917   -0.2921    1.0000         0         0
         0         0         0         0   -0.2697   -0.0861    1.0000         0
         0         0         0         0         0   -0.2948   -0.2931    1.0000
U = 8×8
    4.0000   -1.0000   -1.0000         0         0         0         0         0
         0    3.7500   -0.2500   -1.0000         0         0         0         0
         0         0    3.7333   -1.0667   -1.0000         0         0         0
         0         0         0    3.4286   -0.2857   -1.0000         0         0
         0         0         0         0    3.7083   -1.0833   -1.0000         0
         0         0         0         0         0    3.3919   -0.2921   -1.0000
         0         0         0         0         0         0    3.7052   -1.0861
         0         0         0         0         0         0         0    3.3868
```

Vemos que L es una matriz triangular inferior, entonces, podemos calcular la solución parcial Y usando el algoritmo de sustitución progresiva

```
Y = sustitucion_progresiva(L, b)
```

```
Y = 8×1
    5.0000
   16.2500
    2.3333
   15.0000
    1.8750
   14.9228
   21.7909
   40.7873
```

Dado que la matriz U es una matriz triangular superior, tenemos que utilizar el algoritmo de sustitución regresiva para poder encontrar el valor de X

```
X = sustitucion_regresiva(U, Y)
```

```
X = 8×1
    3.9569
    6.5885
    4.2392
    7.3971
    5.6029
    8.7608
    9.4115
   12.0431
```

Por último, podemos comprobar el valor de X con el comando *linsolve*.

```
X_2 = linsolve(A, b)
```

```
X_2 = 8×1
    3.9569
    6.5885
    4.2392
    7.3971
    5.6029
    8.7608
    9.4115
   12.0431
```

Funciones requeridas

Algoritmo de Doolitle

```
function [L, U] = doolitle(A)
    n = size(A,1);
    L = eye(n);
    U = A;

    for k = 1:n - 1
        for i = k + 1:n
            L(i, k) = U(i, k) / U(k, k);
            for j = k:n
```

```
                U(i, j) = U(i, j) - L(i, k) * U(k, j);
            end
        end
    end
end
```

Algoritmo de sustitución regresiva para matrices triangulares superiores cuadradas

```
function [x] = sustitucion_regresiva(U, b)
    n = size(U, 1);
    x = zeros(n, 1);

    for k = n:-1:1
       x(k) = (b(k) - sum(U(k, k+1:n) * x(k + 1:n))) / U(k, k);
    end
end
```

Algoritmo de sustitución progresiva para matrices triangulares inferiores cuadradas

```
function [x] = sustitucion_progresiva(y, b)
    n = size(y, 1);
    x = zeros(n, 1);

    for k = 1:n
       x(k) = (b(k) - sum(y(k, 1:k - 1) * x(1:k - 1))) / y(k,k);
    end
end
```