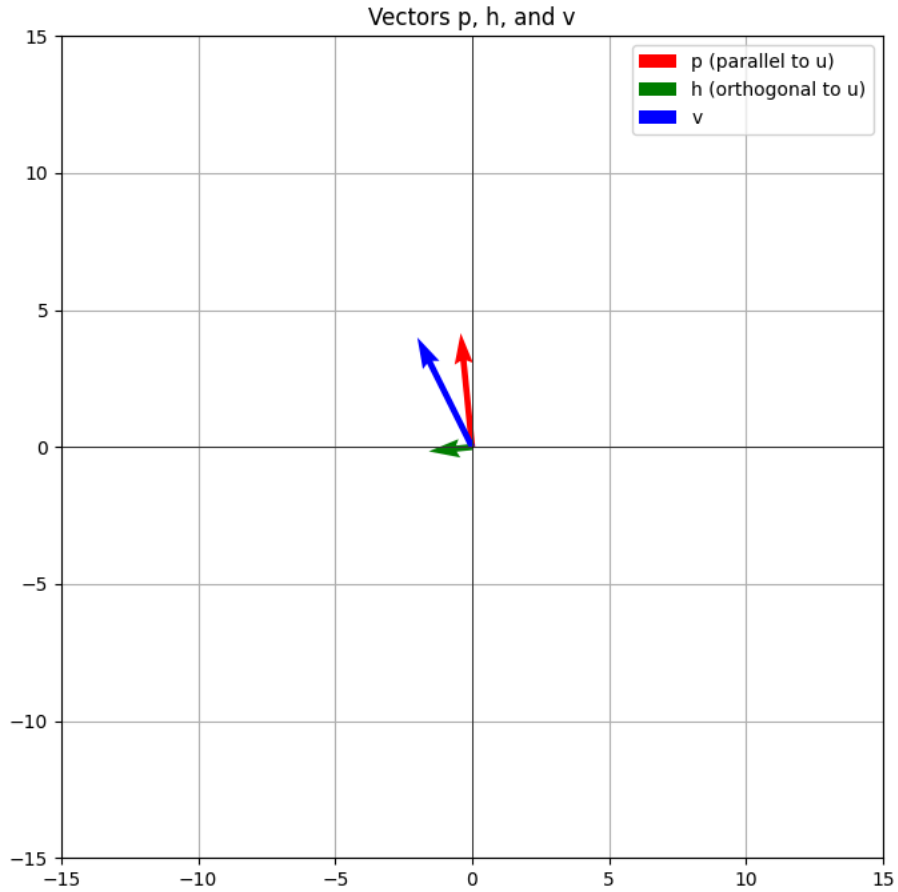
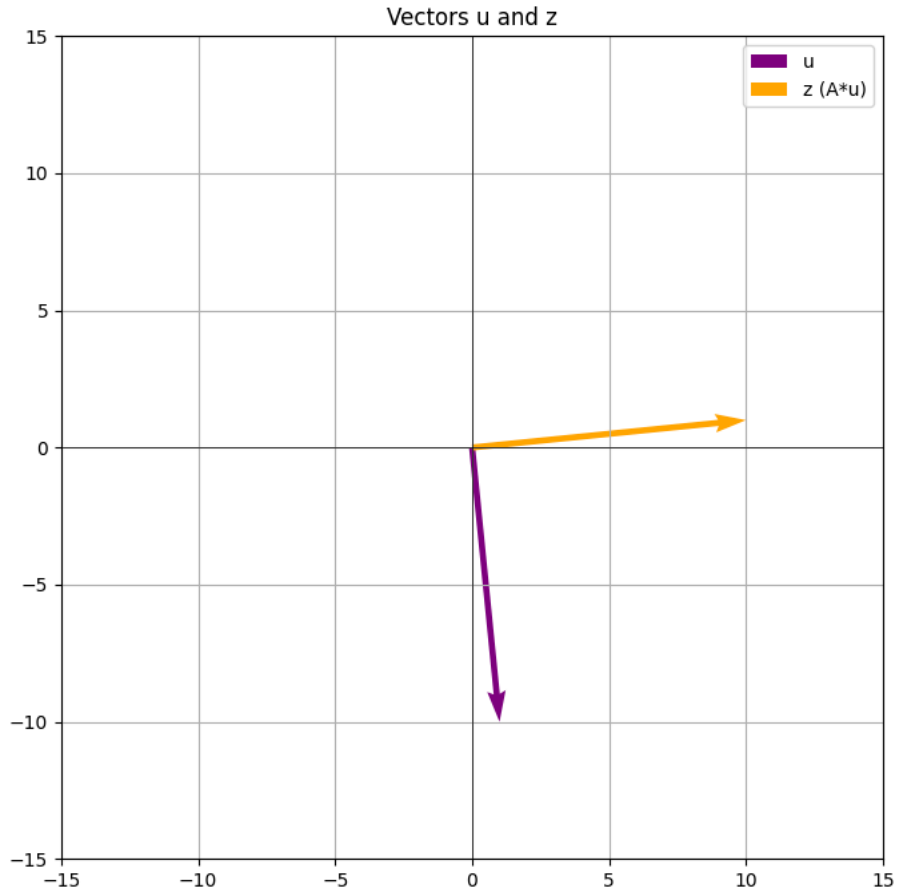
```
#1
import numpy as np

# Definir los vectores
u = np.array([10, 3, 70])
v = np.array([0, -79, -12])
w = np.array([25, 56, 2827])
z = np.array([-10, 0, 58])

# Operación 1
dot_uz = np.dot(u, z)
dot_w3v = np.dot(w, 3*v)
op1 = dot_uz + dot_w3v

# Operación 2
left_term = u + z - v
right_term = z - 3*w + v
op2 = np.dot(left_term, right_term)

# Operación 3
norm_z_w = np.linalg.norm(z - w)
cross_wu = np.cross(w, u)
norm_cross_wu = np.linalg.norm(cross_wu)
op3 = norm_z_w - norm_cross_wu

# Operación 4
norm_u = np.linalg.norm(u)
sum_vz = v + z
part1 = norm_u * sum_vz

norm_2z_3w = np.linalg.norm(2*z - 3*w)
part2 = norm_2z_3w * z

op4 = part1 + part2

#Imprimir
print("Operación 1:", op1)
print("Operación 2:", op2)
print("Operación 3:", op3)
print("Operación 4:", op4)
```

```
Operación 1: -111084
Operación 2: -1201154
Operación 3: -24143.93304782553
Operación 4: [-84380.00472267  -5591.16884023 488554.73592211]
```

```
#2
import numpy as np
import matplotlib.pyplot as plt

u = np.array([1, -10])
v = np.array([-2, 4])


alpha = np.dot(v, u) / np.dot(u, u)
p = alpha * u
h = v - p

#graficar
plt.figure(figsize=(8,8))
plt.quiver(0, 0, p[0], p[1], angles='xy', scale_units='xy', scale=1, color='r', label='p (parallel to u)')
plt.quiver(0, 0, h[0], h[1], angles='xy', scale_units='xy', scale=1, color='g', label='h (orthogonal to u)')
plt.quiver(0, 0, v[0], v[1], angles='xy', scale_units='xy', scale=1, color='b', label='v')

plt.xlim(-15, 15)
plt.ylim(-15, 15)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.grid()
plt.gca().set_aspect('equal', adjustable='box')
plt.legend()
plt.title('Vectors p, h, and v')
plt.show()
```

```
A = np.array([[0, -1], [1, 0]])
z = A @ u

plt.figure(figsize=(8,8))
plt.quiver(0, 0, u[0], u[1], angles='xy', scale_units='xy', scale=1, color='purple', label='u')
plt.quiver(0, 0, z[0], z[1], angles='xy', scale_units='xy', scale=1, color='orange', label='z (A*u)')

plt.xlim(-15, 15)
plt.ylim(-15, 15)
plt.axhline(0, color='black',linewidth=0.5)
plt.axvline(0, color='black',linewidth=0.5)
plt.grid()
plt.gca().set_aspect('equal', adjustable='box')
plt.legend()
plt.title('Vectors u and z')
plt.show()

p, h, z
```






```
(array([-0.41584158,  4.15841584]),
 array([-1.58415842, -0.15841584]),
 array([10,  1]))
```

```
#3
import numpy as np
A = np.array([[1, 2, 4, 5, -2],
              [-4, -4, 3, 6, 0],
              [5, 6, 7, 8, 1],
              [13, 0, -30, 3, 3]])

B = np.array([[2, 5, 0, 15, -1],
              [7, 3, 8, -20, 0],
              [3, 0, 1, 0, 10],
              [4, -5, 2, 10, 9]])

C = np.array([[5, -8, 0, 20, 1],
              [2, -4, -3, 5, 2],
              [-5, 0, 4, 50, 24],
              [0, 90, -5, 3, 4]])
part_a_X = (-3 * A + C - B) / 2
part_b_X = (6 * A - B - 2 * C) / 2

part_a_X, part_b_X
```

```
(array([[  0. ,  -9.5,  -6. ,  -5. ,   4. ],
        [  3.5,   2.5, -10. ,   3.5,   1. ],
        [-11.5,  -9. ,  -9. ,  13. ,   5.5],
        [-21.5,  47.5,  41.5,  -8. ,  -7. ]]),
 array([[ -3. ,  11.5,  12. , -12.5,  -6.5],
        [-17.5,  -9.5,   8. ,  23. ,  -2. ],
        [ 18.5,  18. ,  16.5, -26. , -26. ],
        [ 37. , -87.5, -86. ,   1. ,   0.5]]))
```

```
#4
import numpy as np

# Definimos los valores de x (número de almendros) y y (kilos de almendros)
x_values = np.array([40, 60, 110])
y_values = np.array([20000, 24000, 16500])

A = np.vstack([x_values**2, x_values, np.ones(len(x_values))]).T


a, b, c = np.linalg.solve(A, y_values)

print(f"Coeficientes: a = {a}, b = {b}, c = {c}")

# Punto b: calcular los kilos de almendros para 50 almendros
x = 50
y_pred = a * x**2 + b * x + c
print(f"El número de kilos de almendros para 50 almendros es: {y_pred}")
```

```
Coeficientes: a = -5.0, b = 700.0, c = 0.0
El número de kilos de almendros para 50 almendros es: 22500.0
```

```
#5
import numpy as np

# Definimos las ecuaciones basadas en el promedio de temperaturas
eq1 = "T1 - (0 + T2 + T4 + 20)/4 = 0"
eq2 = "T2 - (20 + T3 + T5 + T1)/4 = 0"
eq3 = "T3 - (20 + 15 + T6 + T2)/4 = 0"
eq4 = "T4 - (0 + T5 + 0 + T1)/4 = 0"
eq5 = "T5 - (T1 + T6 + 0 + T4)/4 = 0"
eq6 = "T6 - (T2 + 15 + 0 + T5)/4 = 0"
A = np.array([
    [4, -1,  0, -1,  0,  0],
    [-1, 4, -1,  0, -1,  0],
    [0, -1,  4,  0,  0, -1],
    [-1,  0,  0,  4, -1,  0],
    [0, -1,  0, -1,  4, -1],
    [0,  0, -1,  0, -1,  4]
])

B = np.array([20, 20, 35, 0, 0, 15])

# Resolvemos el sistema de ecuaciones
temperaturas = np.linalg.solve(A, B)
```

```
# Imprimimos las temperaturas en cada punto
for i, temp in enumerate(temperaturas):
    print(f"T{i+1} = {temp:.2f}°")

# Encontramos los índices de las temperaturas máxima y mínima
max_temp_index = np.argmax(temperaturas)
min_temp_index = np.argmin(temperaturas)

print(f"\nTemperatura máxima en T{max_temp_index + 1} = {temperaturas[max_temp_index]:.2f}°")
print(f"Temperatura mínima en T{min_temp_index + 1} = {temperaturas[min_temp_index]:.2f}°")
```

```
T1 = 9.04°
T2 = 12.33°
T3 = 14.04°
T4 = 3.82°
T5 = 6.24°
T6 = 8.82°

Temperatura máxima en T3 = 14.04°
Temperatura mínima en T4 = 3.82°
```

```
import numpy as np
# Paso 1: Definir la matriz de codificación A (5x5)
A = np.array([
    [0, 0, 1, -1, 0],
    [0, 4, -1, 1, 2],
    [-2, 4, 1, 1, 2],
    [2, -4, 0, 0, -2],
    [0, 2, -1, 1, 2]
])

C_values = [-27, 1, 26, -25, 3, -7, -12, 10, -18, 0, -26, -16, 6, -8, -8, 10,
            149, 91, 94, 193, 15, 51, 148, 22, 120, 114, 48, 38, 146, 44, 56, 2,
            117, 117, 110, 193, -3, 33, 170, 50, 118, 86, -4, 38, 152, 66, 24, 8,
            -88, -90, -82, -166, 38, -2, -126, -4, -92, -76, 34, -12, -102, -18,
            -6, 20, 93, 49, 50, 137, 7, 41, 108, 12, 88, 58, 40, 28, 90, 42, 32, -4]

C = np.array(C_values).reshape(5, 16)

det_A = np.linalg.det(A)

if det_A != 0:
    # Calcular la inversa de A
    A_inv = np.linalg.inv(A)

    B = np.dot(A_inv, C)


    B = np.round(B).astype(int)

    abecedario = {i: chr(96 + i) for i in range(1, 15)}
    abecedario[15] = 'ñ'  # La letra ñ
    abecedario.update({i: chr(96 + i - 1) for i in range(16, 28)})
    abecedario[28] = ' '  # Espacio

    letras = []
    for valor in B.flatten():
        if 1 <= valor <= 28:
            letras.append(abecedario.get(valor, '?'))
        else:
            letras.append('?')

    print("Matriz de codificación A:")
    print(A)

    print("\nMatriz codificada C:")
    print(C)
```