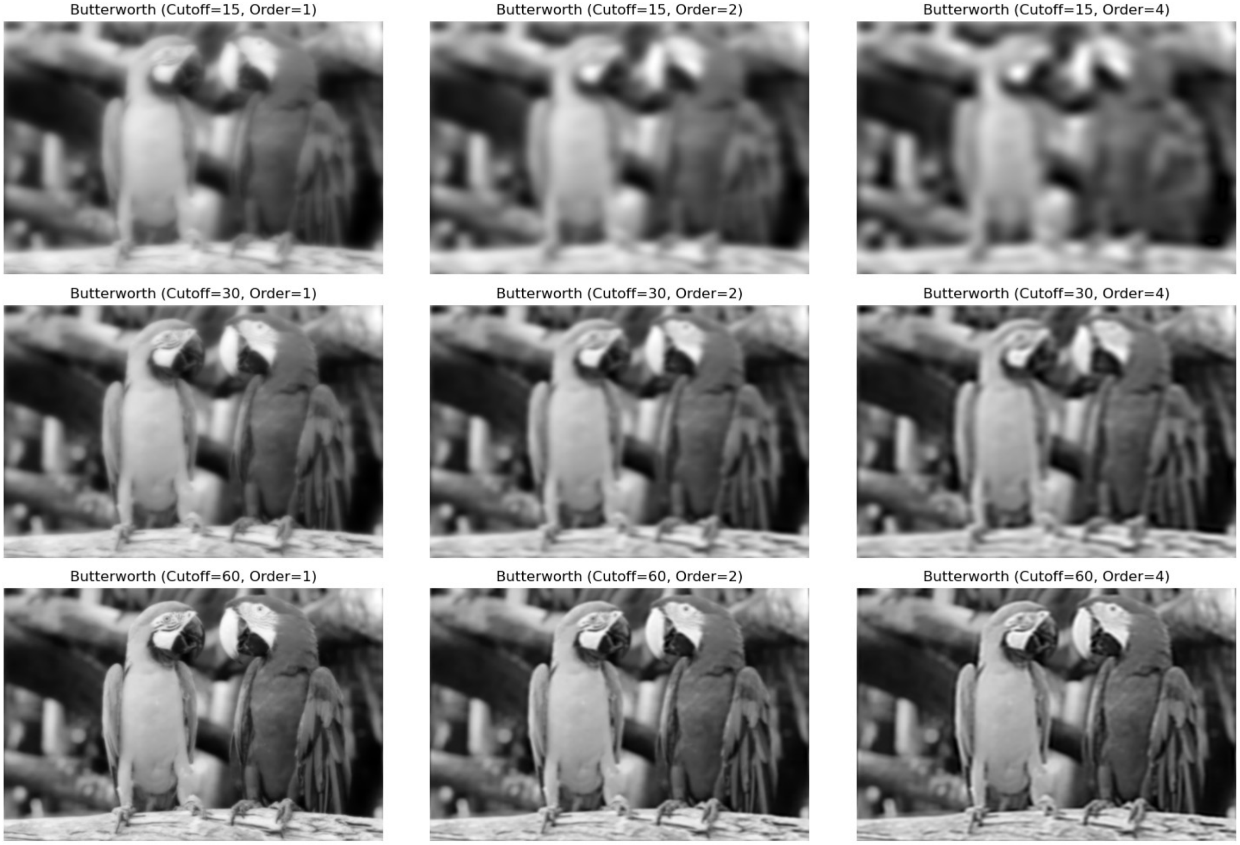
Question 4

```
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Function to apply Fourier Transform and display magnitude spectrum
def apply_fourier_transform(image):
    f_transform = np.fft.fft2(image)
    f_shifted = np.fft.fftshift(f_transform)  # Shift the zero-
frequency component to the center
    magnitude_spectrum = 20 * np.log(np.abs(f_shifted))
    return f_shifted, magnitude_spectrum

# Function to apply inverse Fourier Transform
def inverse_fourier_transform(f_shifted):
    f_ishifted = np.fft.ifftshift(f_shifted)
    img_back = np.fft.ifft2(f_ishifted)
    img_back = np.abs(img_back)
    return img_back

# Butterworth filter
def butterworth_filter(shape, cutoff, order, highpass=True):
    rows, cols = shape
    crow, ccol = rows // 2 , cols // 2
    butter_filter = np.zeros((rows, cols), dtype=np.float32)

    for u in range(rows):
        for v in range(cols):
            d = np.sqrt((u - crow) ** 2 + (v - ccol) ** 2)
            if highpass:
                butter_filter[u, v] = 1 / (1 + (cutoff / d) ** (2 *
order))
            else:
                butter_filter[u, v] = 1 / (1 + (d / cutoff) ** (2 *
order))

    return butter_filter

# Gaussian filter
def gaussian_filter(shape, cutoff, highpass=True):
    rows, cols = shape
    crow, ccol = rows // 2 , cols // 2
    gaussian_filter = np.zeros((rows, cols), dtype=np.float32)

    for u in range(rows):
        for v in range(cols):
            d = np.sqrt((u - crow) ** 2 + (v - ccol) ** 2)
            if highpass:
                gaussian_filter[u, v] = 1 - np.exp(-(d ** 2) / (2 *
(cutoff ** 2)))
            else:
                gaussian_filter[u, v] = np.exp(-(d ** 2) / (2 *
(cutoff ** 2)))

    return gaussian_filter

# Load an image (grayscale)
image_path = 'image_bmp.bmp'  # Replace with your image path
image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

# Apply Fourier Transform
f_shifted, magnitude_spectrum = apply_fourier_transform(image)

# Experiment with different Butterworth cutoff frequencies and orders
cutoff_values = [15, 30, 60]  # Different cutoff frequencies
order_values = [1, 2, 4]      # Different filter orders

# Experiment with different Gaussian cutoff frequencies
gaussian_cutoffs = [15, 30, 60]  # Different cutoff frequencies

# Plot the results for Butterworth filters with different parameters
plt.figure(figsize=(15, 10))

for i, cutoff in enumerate(cutoff_values):
    for j, order in enumerate(order_values):
        butter_filter = butterworth_filter(image.shape, cutoff=cutoff,
order=order, highpass=False)
        butter_filtered = f_shifted * butter_filter
        butter_result = inverse_fourier_transform(butter_filtered)

        plt.subplot(3, 3, i * 3 + j + 1)
        plt.imshow(butter_result, cmap='gray')
        plt.title(f'Butterworth (Cutoff={cutoff}, Order={order})')
        plt.axis('off')

plt.tight_layout()
plt.show()

# Plot the results for Gaussian filters with different cutoff
frequencies
plt.figure(figsize=(12, 6))

for i, cutoff in enumerate(gaussian_cutoffs):
    gaussian_filter_result = gaussian_filter(image.shape,
cutoff=cutoff, highpass=False)
    gaussian_filtered = f_shifted * gaussian_filter_result
    gaussian_result = inverse_fourier_transform(gaussian_filtered)
    plt.subplot(1, 3, i + 1)
    plt.imshow(gaussian_result, cmap='gray')
    plt.title(f'Gaussian (Cutoff={cutoff})')
    plt.axis('off')

plt.tight_layout()
plt.show()
```


Butterworth (Cutoff=15, Order=1)
Butterworth (Cutoff=15, Order=2)
Butterworth (Cutoff=15, Order=4)
Butterworth (Cutoff=30, Order=1)
Butterworth (Cutoff=30, Order=2)
Butterworth (Cutoff=30, Order=4)
Butterworth (Cutoff=60, Order=1)
Butterworth (Cutoff=60, Order=2)
Butterworth (Cutoff=60, Order=4)