In [1]:
```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.optimize import minimize
from scipy import signal

Z_ALPHA = 2.576
BURN_IN = 20

def penalty(upper_bound, lower_bound):
    return (upper_bound-lower_bound)**2
```

데이터 생성

In [2]:
```python
def split_segment(n, n_bkps, min_seg):
    max_num = n - min_seg * n_bkps
    result = []

    for i in range(n_bkps):
        if i == n_bkps - 1:
            seg_size = n - sum(result)
        else:
            num = int(round(np.random.uniform(0, max_num)))
            seg_size = min_seg + num
            max_num = max(0, max_num - num)

        result.append(seg_size)

    return result

def generate_custom_data(n, segment_means, segment_vars, min_seg=30):
    n_bkps = len(segment_means)

    if len(segment_means) != len(segment_vars):
        raise ValueError("segment_means and segment_vars must have the same length"

    segment_sizes = split_segment(n, n_bkps, min_seg)

    signal = []
    bkps = []
    for i in range(n_bkps):
        segment_data = np.random.normal(segment_means[i], np.sqrt(segment_vars[i]),
        signal.extend(segment_data)
        bkps.append(sum(segment_sizes[:i+1]))

    return {"signal": np.array(signal), "means": segment_means, "variances": segmen

def display(title, signal, bkps, display_mean=False, points=[]):
    sns.set(style="whitegrid")
    plt.figure(figsize=(20, 5))

    # Plot signal data
    plt.plot(signal, label="Signal", color='black', linewidth=1)
```

```
    plt.title(title, fontsize=20, fontweight='bold')

    # Plot change points with increased transparency
    for i in range(len(bkps)):
        plt.axvspan(bkps[i-1] if i > 0 else 0, bkps[i], color=sns.color_palette("hu

    if display_mean:
        start = 0
        for i in range(len(bkps)):
            end = bkps[i]
            mean_value = np.mean(signal[start:end])
            plt.plot(range(start, end), [mean_value] * (end - start), 'b--', linewi
            start = end

    if len(points) > 0:
        plt.scatter(points, signal[points], color='red', s=50, label='Points')

    for m in bkps:
        if m < len(signal):
            plt.axvline(x=m, color='red', linestyle='--', label=f'Change Point (m={

    plt.legend()
    plt.show()

    # Print segment means and variances
    print("Segment Means and Variances:")
    start = 0
    for i in range(len(bkps)):
        end = bkps[i]
        segment_data = signal[start:end]
        segment_mean = np.mean(segment_data)
        segment_variance = np.var(segment_data)
        print(f"Segment {i+1}: Mean = {segment_mean:.2f}, Variance = {segment_varia
        start = end
```

Plot 그리기

In [3]:
```
def plot_Lk_over_k(signal, bkps, min_k=2):
    n = len(signal)
    k_values = np.arange(min_k, n + 1)
    E_Lk_over_k_minus_1 = np.zeros(n - min_k + 1)
    std_dev = np.zeros(n - min_k + 1)

    for k in k_values:
        Lk = signal[:k]
        if len(Lk) >= (k - 1):
            E_Lk = np.mean(Lk)
            E_Lk_over_k_minus_1[k - min_k] = E_Lk / (k - 1)
            std_dev[k - min_k] = np.std(Lk) / (k - 1)

    # Ensure we handle cases where E_Lk_over_k_minus_1 or std_dev might not match k
    valid_indices = np.where(E_Lk_over_k_minus_1 != 0)[0]
    k_values = k_values[valid_indices]
    E_Lk_over_k_minus_1 = E_Lk_over_k_minus_1[valid_indices]
    std_dev = std_dev[valid_indices]
```

```python
    # Confidence intervals
    lower_bound = E_Lk_over_k_minus_1 - Z_ALPHA * std_dev
    upper_bound = E_Lk_over_k_minus_1 + Z_ALPHA * std_dev

    plt.figure(figsize=(20, 5))

    # Plot E(L_k / (k - 1))
    plt.plot(k_values, E_Lk_over_k_minus_1, label=r'$E\left(\frac{L_k}{k-1}\right)$

    # Plot confidence intervals
    plt.fill_between(k_values, lower_bound, upper_bound, color='gray', alpha=0.3, l

    for m in bkps:
        if m < len(signal):
            plt.axvline(x=m, color='red', linestyle='--', label=f'Change Point (m={

    plt.xlabel('k')
    plt.ylabel(r'$E\left(\frac{L_k}{k-1}\right)$')
    plt.title(r'$E\left(\frac{L_k}{k-1}\right)$ with Confidence Interval', fontsize
    plt.legend()
    plt.grid(True)
    plt.show()

def find_local_minima(x, y):
    # Find the indices of local minima
    minima_indices = signal.argrelextrema(y, np.less)[0]

    if len(minima_indices) == 0:
        return np.array([]), np.array([])

    minima_x = x[minima_indices]  # 극소점의 x값
    minima_y = y[minima_indices]  # 극소점의 y값

    return minima_x, minima_y

def plot_Lk_over_k_with_minima(signal, bkps, min_k=2):
    n = len(signal)
    k_values = np.arange(min_k, n + 1)
    E_Lk_over_k_minus_1 = np.zeros(n - min_k + 1)
    std_dev = np.zeros(n - min_k + 1)

    for k in k_values:
        Lk = signal[:k]
        if len(Lk) >= (k - 1):
            E_Lk = np.mean(Lk)
            E_Lk_over_k_minus_1[k - min_k] = E_Lk / (k - 1)
            std_dev[k - min_k] = np.std(Lk) / (k - 1)

    # Ensure we handle cases where E_Lk_over_k_minus_1 or std_dev might not match k
    valid_indices = np.where(E_Lk_over_k_minus_1 != 0)[0]
    k_values = k_values[valid_indices]
    E_Lk_over_k_minus_1 = E_Lk_over_k_minus_1[valid_indices]
    std_dev = std_dev[valid_indices]

    # Confidence intervals
```

```
    lower_bound = E_Lk_over_k_minus_1 - 2.576 * std_dev
    upper_bound = E_Lk_over_k_minus_1 + 2.576 * std_dev

    new_E_Lk = E_Lk_over_k_minus_1 + penalty(upper_bound, lower_bound)

    # Find local minima
    minima_k, minima_values = find_local_minima(k_values, new_E_Lk)
    min_index = np.argmin(minima_values)
    min_x = minima_k[min_index]
    min_y = minima_values[min_index]

    plt.figure(figsize=(20, 5))

    # Plot E(L_k / (k - 1))
    plt.plot(k_values, new_E_Lk, label=r'$E\left(\frac{L_k}{k-1}\right)$', color='b

    plt.scatter(minima_k, minima_values, color='red', zorder=5, label=f'Local Minim
    plt.scatter(min_x, min_y, color='blue', zorder=5, label=f'Detected CP (k={min_x

    for m in bkps:
        if m < len(signal):
            plt.axvline(x=m, color='red', linestyle='--', label=f'Change Point (m={

    plt.xlabel('k')
    plt.ylabel(r'$E\left(\frac{L_k}{k-1}\right)$')
    plt.title(r'$E\left(\frac{L_k}{k-1}\right)$ + Penalty with Local Minima', fonts
    plt.legend()
    plt.grid(True)
    plt.show()
```

데이터 변환 (패널티 더하기)

In [4]:

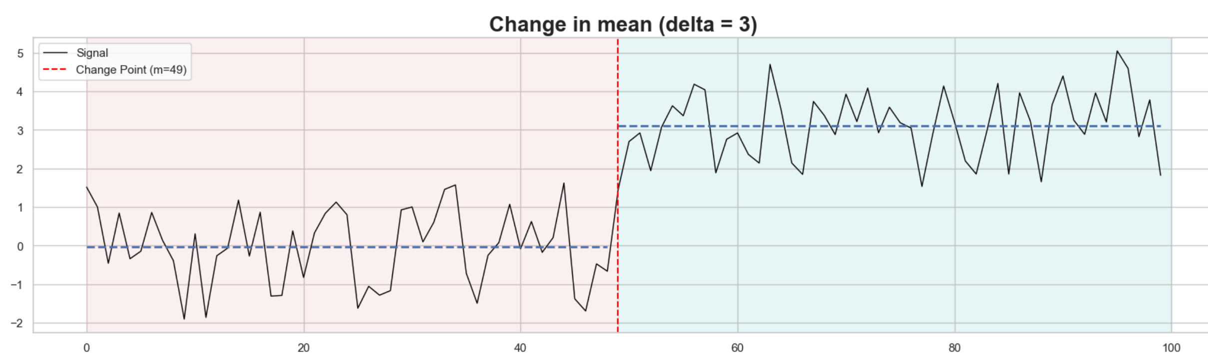
```
def transform_data(signal, min_k=2):
    n = len(signal)
    k_values = np.arange(min_k, n + 1)
    E_Lk_over_k_minus_1 = np.zeros(n - min_k + 1)
    std_dev = np.zeros(n - min_k + 1)

    for k in k_values:
        Lk = signal[:k]
        if len(Lk) >= (k - 1):
            E_Lk = np.mean(Lk)
            E_Lk_over_k_minus_1[k - min_k] = E_Lk / (k - 1)
            std_dev[k - min_k] = np.std(Lk) / (k - 1)

    valid_indices = np.where(E_Lk_over_k_minus_1 != 0)[0]
    k_values = k_values[valid_indices]
    E_Lk_over_k_minus_1 = E_Lk_over_k_minus_1[valid_indices]
    std_dev = std_dev[valid_indices]

    lower_bound = E_Lk_over_k_minus_1 - Z_ALPHA * std_dev
    upper_bound = E_Lk_over_k_minus_1 + Z_ALPHA * std_dev

    new_E_Lk = E_Lk_over_k_minus_1 + (upper_bound-lower_bound)**2
```

```
    return k_values, new_E_Lk
```

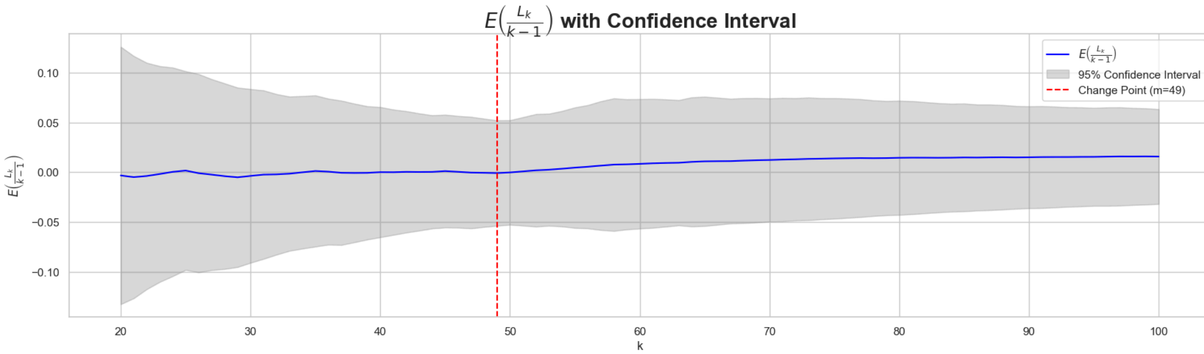
Change Point 찾기

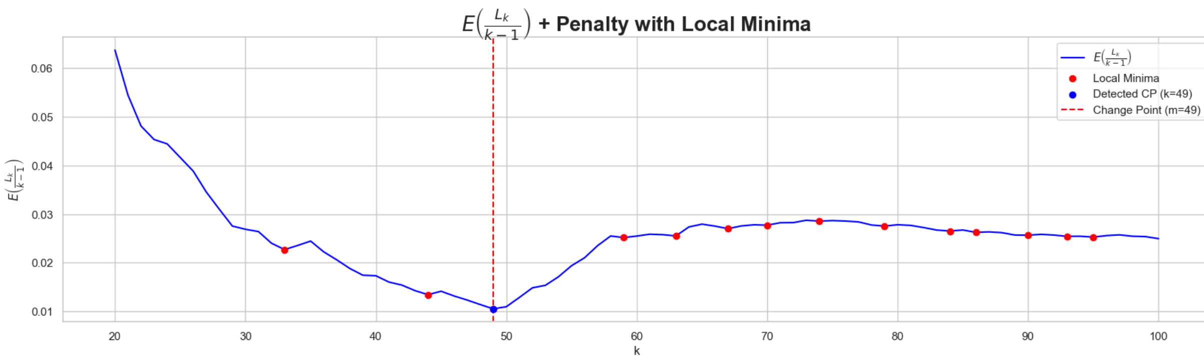
In [5]:
```
def find_change_point(signal):
    k_values, E_Lk = transform_data(signal)
    change_points = []
    n = len(signal)

    for k in range(10, n - 10):
        slope1 = (E_Lk[k] - E_Lk[k - 10]) / 10
        slope2 = (E_Lk[k + 9] - E_Lk[k]) / 10

        if slope1 < 0 and slope2 > 0:
            change_points.append(k)

    return change_points
```

In [6]:
```
# Example usage
n = 100  # Total number of data points
segment_means = [0, 3]  # Means for each segment
segment_vars = [1, 1]  # Variances for each segment

data = generate_custom_data(n, segment_means, segment_vars)
display('Change in mean (delta = 3)', data['signal'], data['bkps'], display_mean=Tr
plot_Lk_over_k(data['signal'], data['bkps'], BURN_IN)

plot_Lk_over_k_with_minima(data['signal'], data['bkps'], BURN_IN)

cp = find_change_point(data['signal'])
```



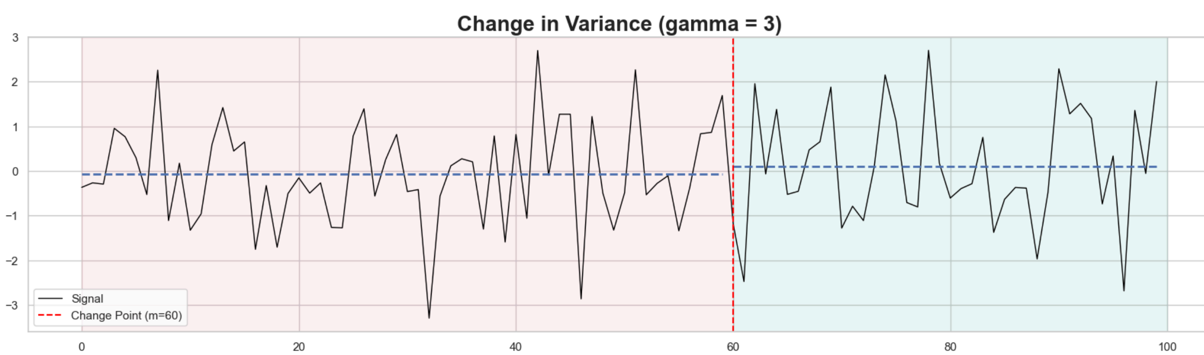

```
Segment Means and Variances:
Segment 1: Mean = -0.04, Variance = 0.97
Segment 2: Mean = 3.11, Variance = 0.78
```



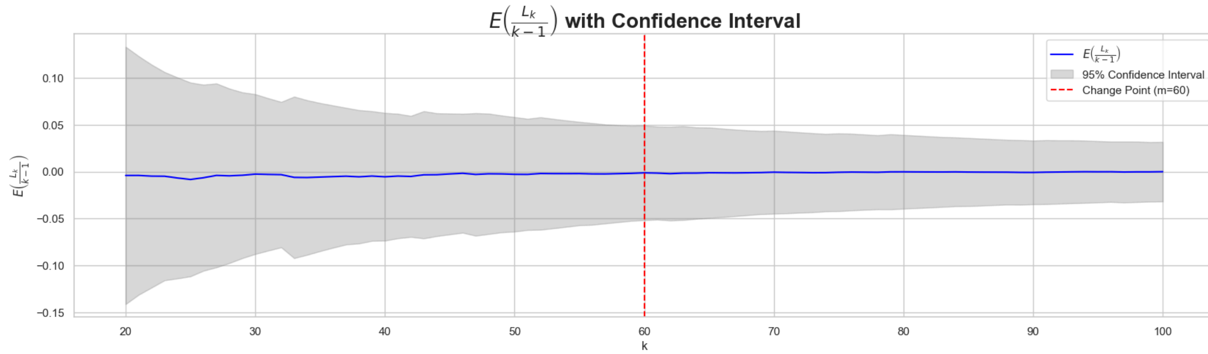


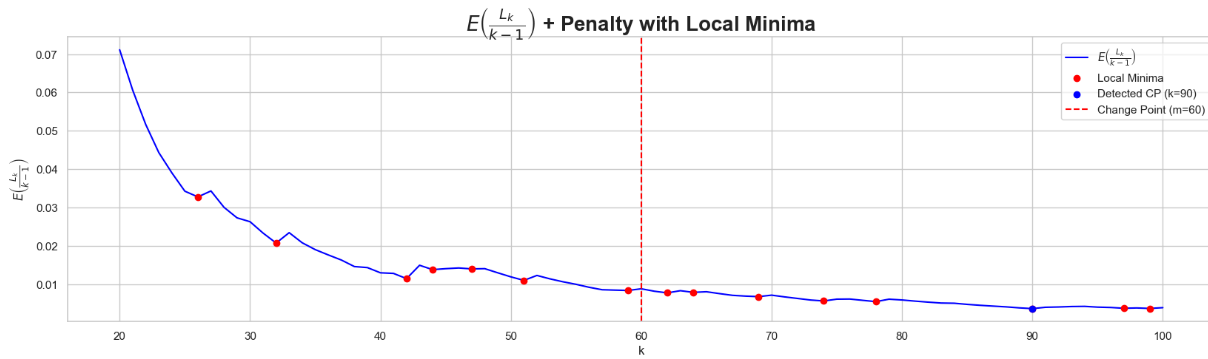
In [7]:

```
# Example usage
n = 100  # Total number of data points
segment_means = [0, 0]  # Means for each segment
segment_vars = [1, 3]  # Variances for each segment

data = generate_custom_data(n, segment_means, segment_vars)
display('Change in Variance (gamma = 3)', data['signal'], data['bkps'], display_mea
plot_Lk_over_k(data['signal'], data['bkps'], BURN_IN)

plot_Lk_over_k_with_minima(data['signal'], data['bkps'], BURN_IN)

cp = find_change_point(data['signal'])
```

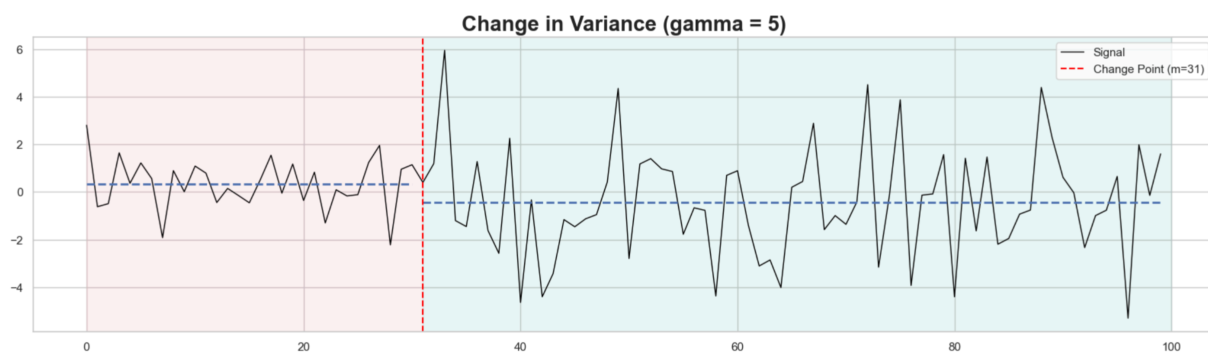



```
Segment Means and Variances:
Segment 1: Mean = -0.08, Variance = 1.33
Segment 2: Mean = 0.10, Variance = 1.69
```

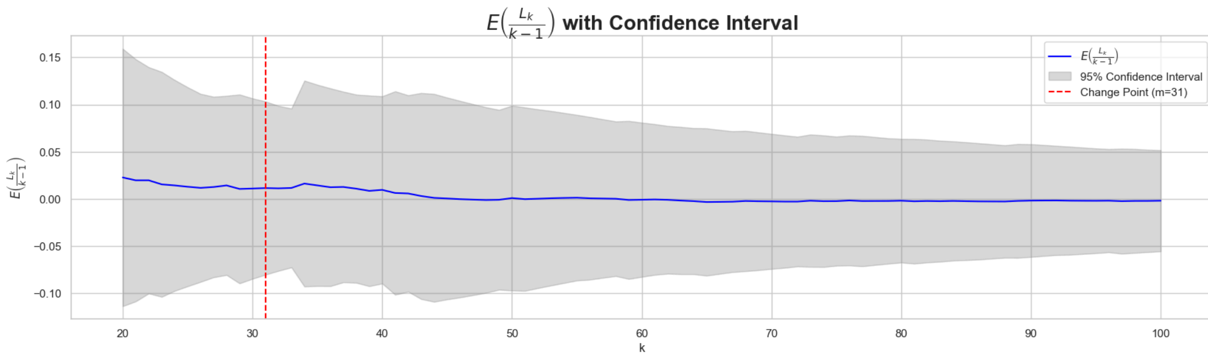


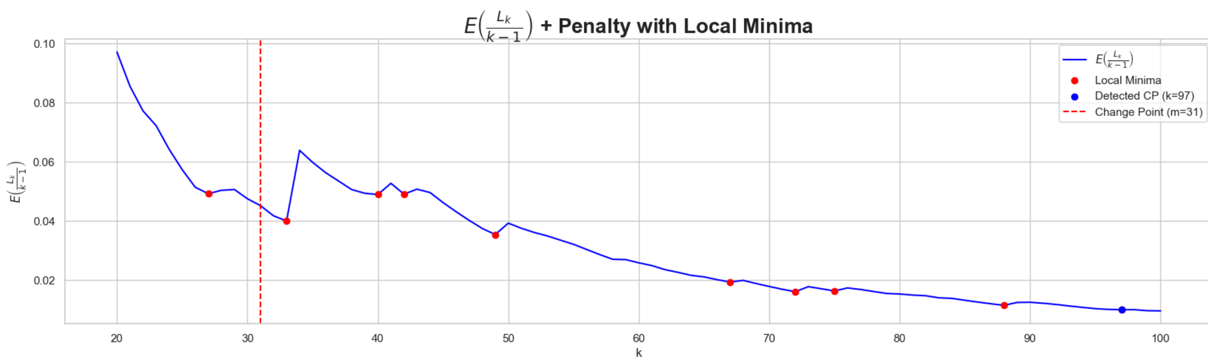


In [8]:
```
# Example usage
n = 100  # Total number of data points
segment_means = [0, 0]  # Means for each segment
segment_vars = [1, 5]  # Variances for each segment

data = generate_custom_data(n, segment_means, segment_vars)
display('Change in Variance (gamma = 5)', data['signal'], data['bkps'], display_mea
plot_Lk_over_k(data['signal'], data['bkps'], BURN_IN)

plot_Lk_over_k_with_minima(data['signal'], data['bkps'], BURN_IN)

cp = find_change_point(data['signal'])
```

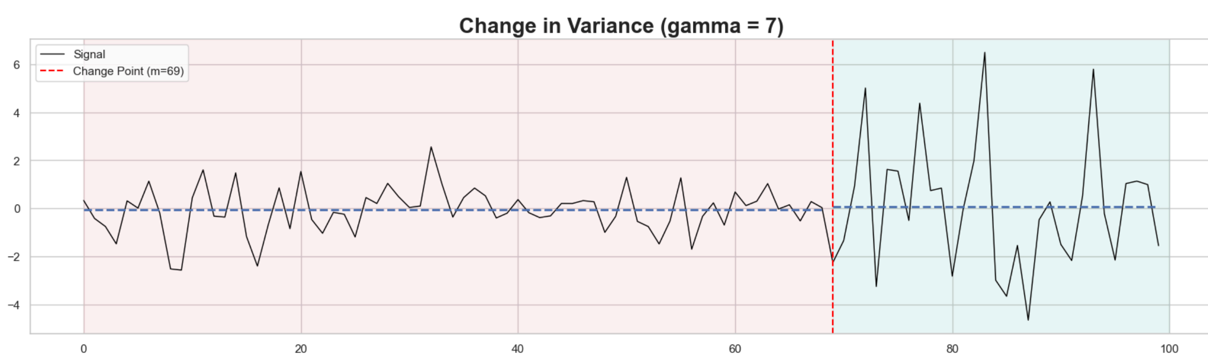



```
Segment Means and Variances:
Segment 1: Mean = 0.34, Variance = 1.14
Segment 2: Mean = -0.43, Variance = 5.45
```

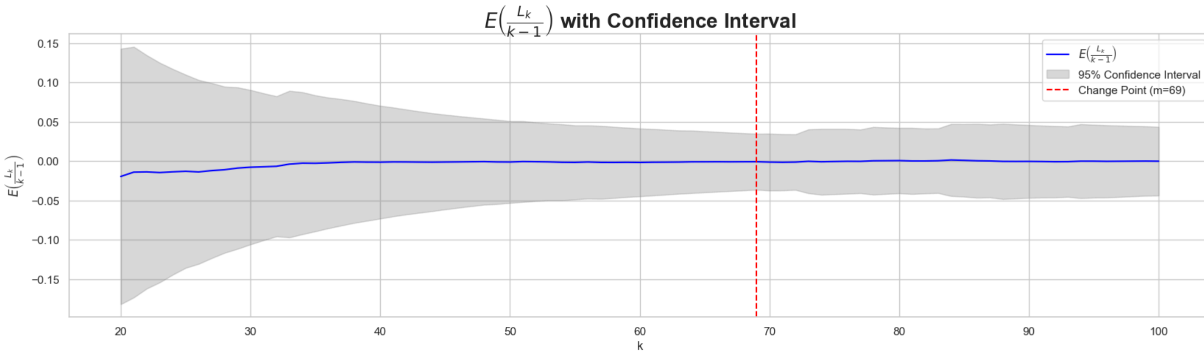


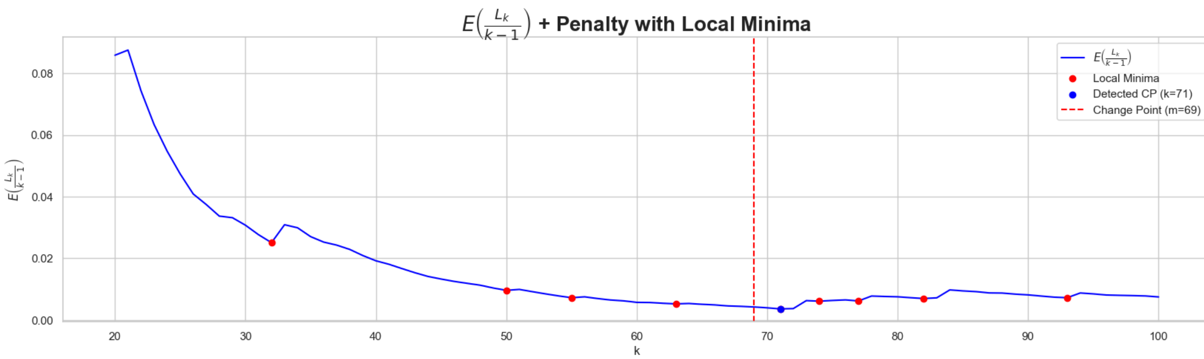


In [9]:
```
# Example usage
n = 100  # Total number of data points
segment_means = [0, 0]  # Means for each segment
segment_vars = [1, 7]  # Variances for each segment

data = generate_custom_data(n, segment_means, segment_vars)
display('Change in Variance (gamma = 7)', data['signal'], data['bkps'], display_mea
plot_Lk_over_k(data['signal'], data['bkps'], BURN_IN)

plot_Lk_over_k_with_minima(data['signal'], data['bkps'], BURN_IN)

cp = find_change_point(data['signal'])
```



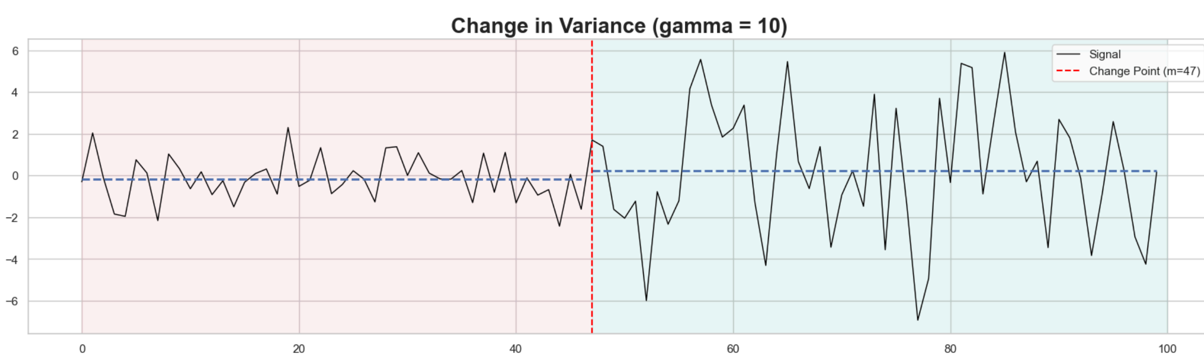

```
Segment Means and Variances:
Segment 1: Mean = -0.06, Variance = 0.88
Segment 2: Mean = 0.08, Variance = 7.10
```



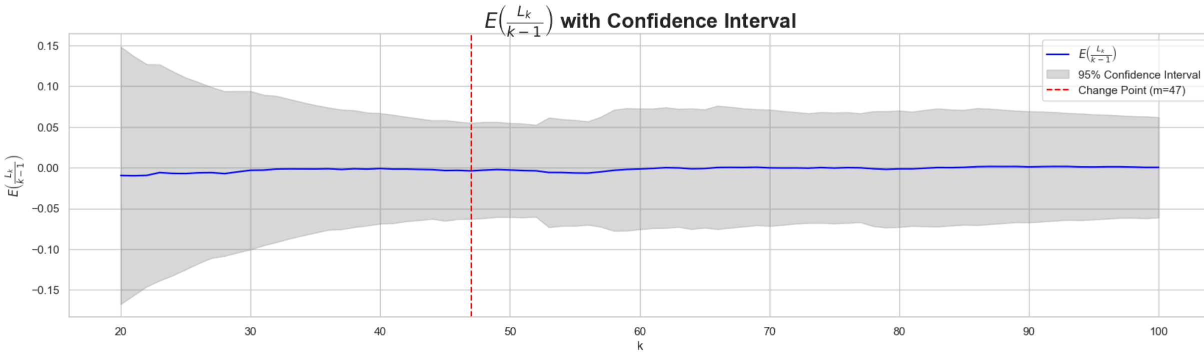


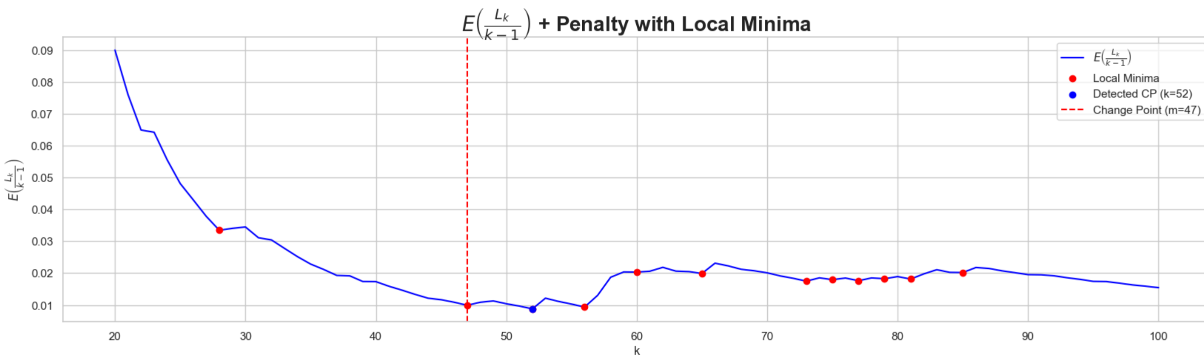
In [10]:

```
# Example usage
n = 100  # Total number of data points
segment_means = [0, 0]  # Means for each segment
segment_vars = [1, 10]  # Variances for each segment

data = generate_custom_data(n, segment_means, segment_vars)
display('Change in Variance (gamma = 10)', data['signal'], data['bkps'], display_me
plot_Lk_over_k(data['signal'], data['bkps'], BURN_IN)

plot_Lk_over_k_with_minima(data['signal'], data['bkps'], BURN_IN)

cp = find_change_point(data['signal'])
```

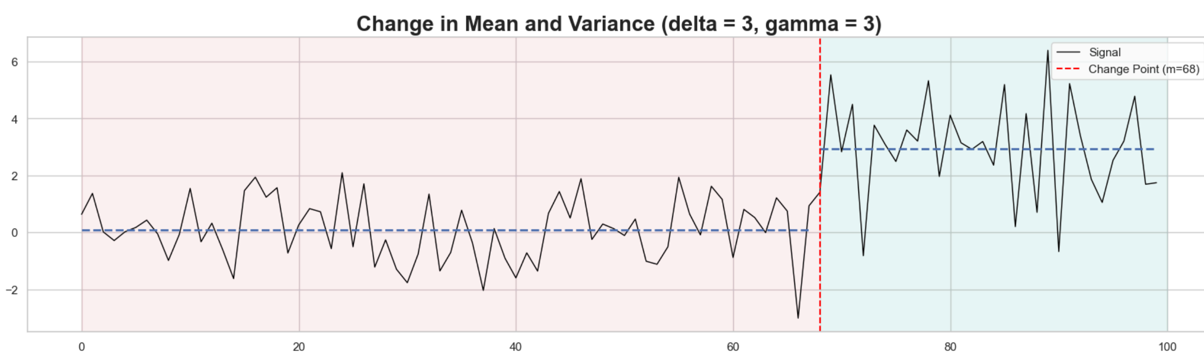



```
Segment Means and Variances:
Segment 1: Mean = -0.18, Variance = 1.11
Segment 2: Mean = 0.22, Variance = 9.49
```

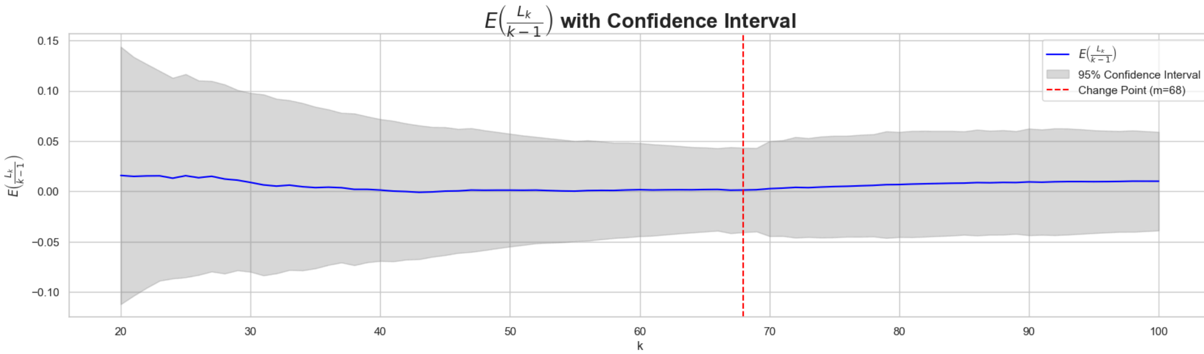




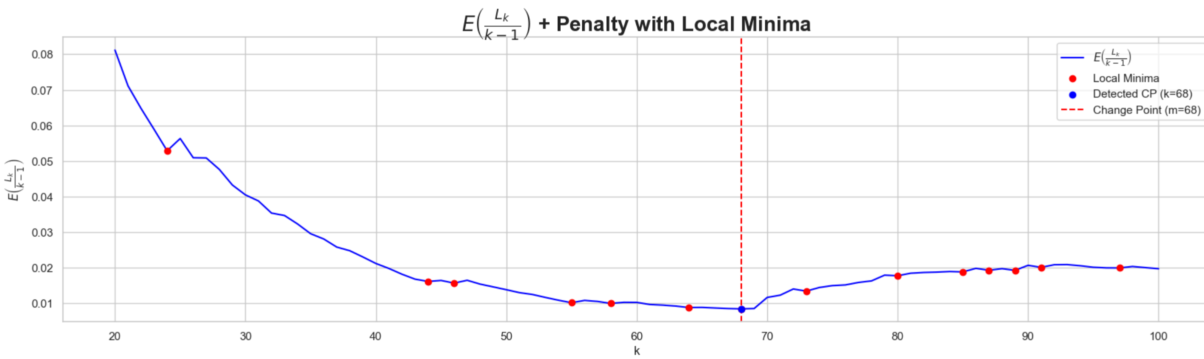
In [11]:
```
# Example usage
n = 100  # Total number of data points
segment_means = [0, 3]  # Means for each segment
segment_vars = [1, 3]  # Variances for each segment

data = generate_custom_data(n, segment_means, segment_vars)
display('Change in Mean and Variance (delta = 3, gamma = 3)', data['signal'], data[
plot_Lk_over_k(data['signal'], data['bkps'], BURN_IN)

plot_Lk_over_k_with_minima(data['signal'], data['bkps'], BURN_IN)

cp = find_change_point(data['signal'])
```




```
Segment Means and Variances:
Segment 1: Mean = 0.09, Variance = 1.19
Segment 2: Mean = 2.94, Variance = 3.01
```



$E\left(\frac{L_k}{k-1}\right)$ with Confidence Interval
$E\left(\frac{L_k}{k-1}\right)$
95% Confidence Interval
Change Point (m=68)
k



$E\left(\frac{L_k}{k-1}\right)$ + Penalty with Local Minima
$E\left(\frac{L_k}{k-1}\right)$
Local Minima
Detected CP (k=68)
Change Point (m=68)
k

In [ ]:

In [ ]: