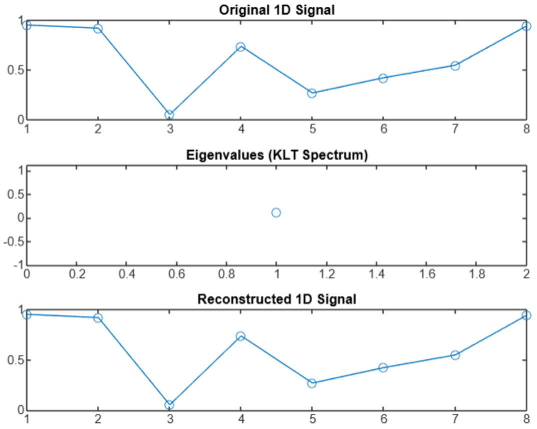
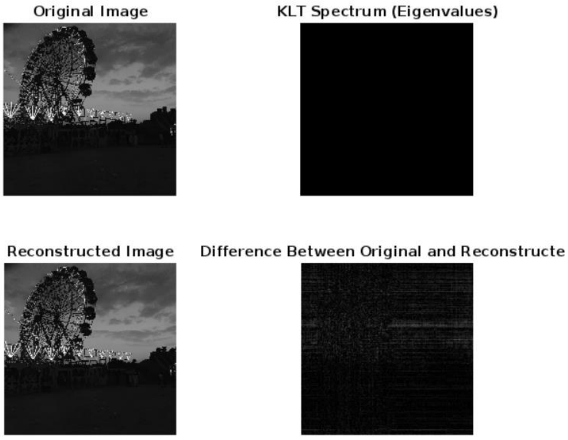
PROGRAM 18: Compute KLT Kernel of Various Sizes and Apply to KLT of Different Signals (1D, 2D).

```
N = [4, 8, 16];

for i = 1:length(N)
   n = N(i);
   X = randn(n);
   C = cov(X);
   [E, D] = eig(C);

   fprintf('KLT Kernel of size %d:\n', n);
   disp(E); % Display KLT Kernel
end

x_1D = rand(1, 8);
Cx = cov(x_1D);
[E_1D, D_1D] = eig(Cx);

x_1D_KLT = E_1D' * x_1D(:);
x_1D_Reconstructed = E_1D * x_1D_KLT;

figure;
subplot(3,1,1); plot(x_1D, '-o'); title('Original 1D Signal');
subplot(3,1,2); plot(diag(D_1D), '-o'); title('Eigenvalues (KLT Spectrum)');
subplot(3,1,3); plot(x_1D_Reconstructed, '-o'); title('Reconstructed 1D Signal');

I_2D = imread('download.jpg');
if size(I_2D, 3) == 3
   I_2D = rgb2gray(I_2D);
end

I_2D = double(I_2D);
[m, n] = size(I_2D);
new_size = min(m, n);

I_2D = imresize(I_2D, [new_size, new_size]);
C_2D = cov(I_2D);

[E_2D, D_2D] = eig(C_2D);

I_2D_KLT = E_2D' * I_2D * E_2D;

I_2D_Reconstructed = E_2D * I_2D_KLT * E_2D';

figure;
subplot(2,2,1); imshow(I_2D, []); title('Original Image');
subplot(2,2,2); imshow(log(1 + abs(D_2D)), []); title('KLT Spectrum (Eigenvalues)');
subplot(2,2,3); imshow(I_2D_Reconstructed, []); title('Reconstructed Image');
subplot(2,2,4); imshow(abs(I_2D - I_2D_Reconstructed), []); title('Difference Between
Original and Reconstructed');
```

OUTPUT:

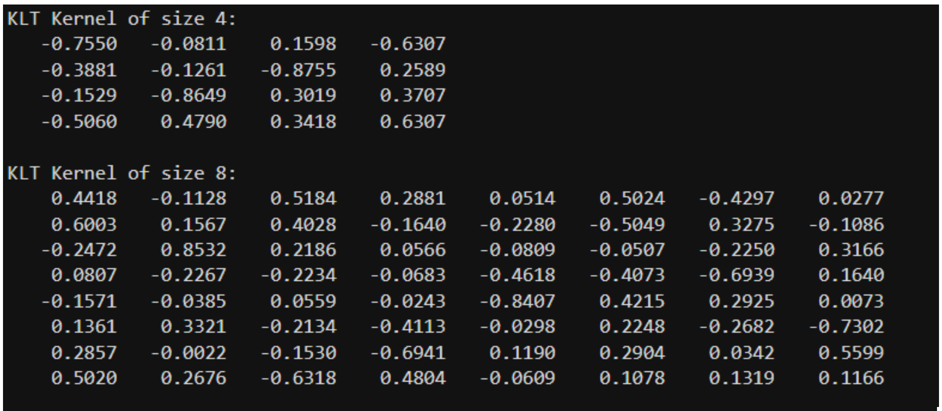



Original Image
KLT Spectrum (Eigenvalues)
Reconstructed Image
Difference Between Original and Reconstructe

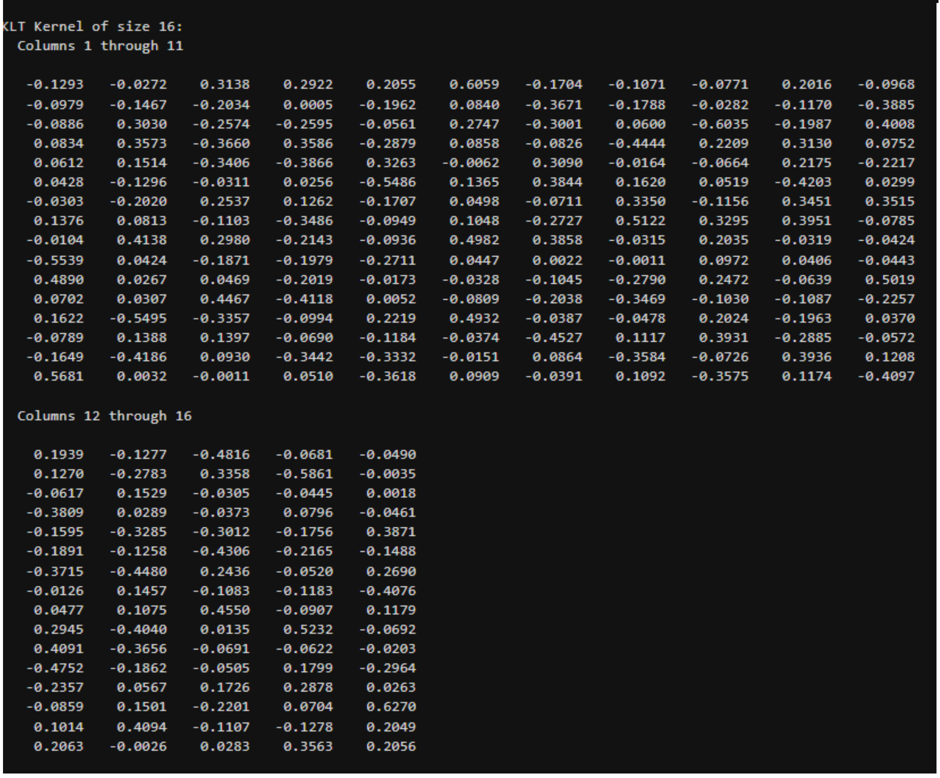
PROGRAM 19: Compute KLT of Different Signals (1D, 2D) [Image] using MATLAB function klt() /inv(klt()).

```
N = [4, 8, 16];

for i = 1:length(N)
   n = N(i);
   X = randn(n, n);
   C = cov(X);
   [E, D] = eig(C);

   fprintf('KLT Kernel of size %d:\n', n);
   disp(E);
end

x_1D = rand(8, 1);
Cx = cov(x_1D);
[E_1D, D_1D] = eig(Cx);
x_1D_KLT = E_1D' * x_1D;
x_1D_Reconstructed = E_1D * x_1D_KLT;

figure;
subplot(3,1,1); plot(x_1D, '-o'); title('Original 1D Signal');
subplot(3,1,2); plot(diag(D_1D), '-o'); title('KLT Spectrum (Eigenvalues)');
subplot(3,1,3); plot(x_1D_Reconstructed, '-o'); title('Reconstructed 1D Signal');

I_2D = imread('download.jpg');

if size(I_2D, 3) == 3
   I_2D = rgb2gray(I_2D);
end

I_2D = double(I_2D);
[m, n] = size(I_2D);
new_size = min(m, n);
I_2D = imresize(I_2D, [new_size, new_size]);
C_2D = cov(I_2D);
[E_2D, D_2D] = eig(C_2D);
I_2D_KLT = E_2D' * I_2D * E_2D;
I_2D_Reconstructed = E_2D * I_2D_KLT * E_2D';

figure;
subplot(2,2,1); imshow(I_2D, []); title('Original Image');
subplot(2,2,2); imshow(log(1 + abs(D_2D)), []); title('KLT Spectrum (Eigenvalues)');
subplot(2,2,3); imshow(I_2D_Reconstructed, []); title('Reconstructed Image');
subplot(2,2,4); imshow(abs(I_2D - I_2D_Reconstructed), []); title('Difference Between
Original and Reconstructed');
```

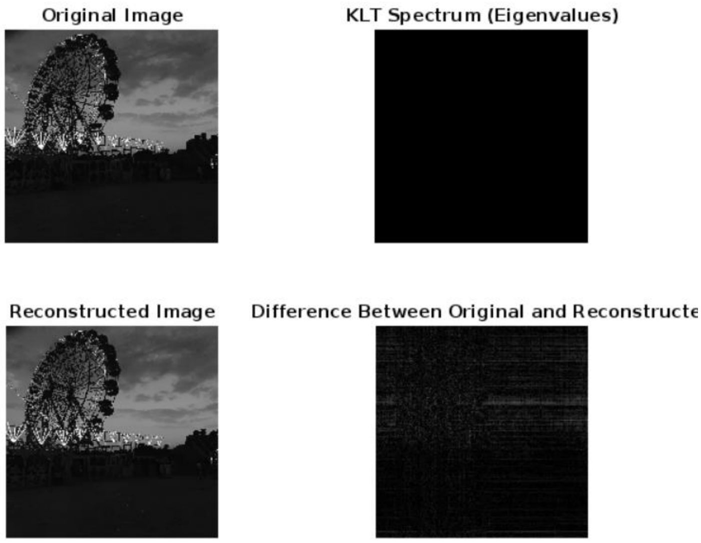
OUTPUT:

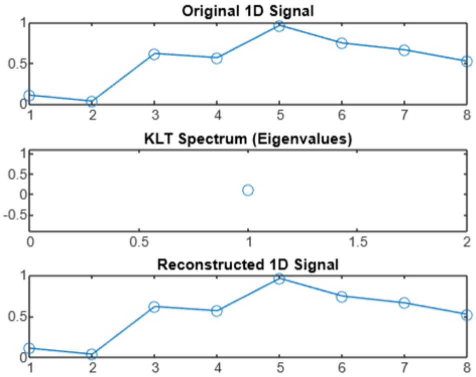


```
KLT Kernel of size 4:
   -0.7550   -0.0811    0.1598   -0.6307
   -0.3881   -0.1261   -0.8755    0.2589
   -0.1529   -0.8649    0.3019    0.3707
   -0.5060    0.4790    0.3418    0.6307

KLT Kernel of size 8:
    0.4418   -0.1128    0.5184    0.2881    0.0514    0.5024   -0.4297    0.0277
    0.6003    0.1567    0.4028   -0.1640   -0.2280   -0.5049    0.3275   -0.1086
   -0.2472    0.8532    0.2186    0.0566   -0.0809   -0.0507   -0.2250    0.3166
    0.0807   -0.2267   -0.2234   -0.0683   -0.4618   -0.4073   -0.6939    0.1640
   -0.1571   -0.0385    0.0559   -0.0243   -0.8407    0.4215    0.2925    0.0073
    0.1361    0.3321   -0.2134   -0.4113   -0.0298    0.2248   -0.2682   -0.7302
    0.2857   -0.0022   -0.1530   -0.6941    0.1190    0.2904    0.0342    0.5599
    0.5020    0.2676   -0.6318    0.4804   -0.0609    0.1078    0.1319    0.1166
```



```
KLT Kernel of size 16:
  Columns 1 through 11

   -0.1293   -0.0272    0.3138    0.2922    0.2055    0.6059   -0.1704   -0.1071   -0.0771    0.2016   -0.0968
   -0.0979   -0.1467   -0.2034    0.0005   -0.1962    0.0840   -0.3671   -0.1788   -0.0282   -0.1170   -0.3885
   -0.0886    0.3030   -0.2574   -0.2595   -0.0561    0.2747   -0.3001    0.0600   -0.6035   -0.1987    0.4008
    0.0834    0.3573   -0.3660    0.3586   -0.2879    0.0858   -0.0826   -0.4444    0.2209    0.3130    0.0752
    0.0612    0.1514   -0.3406   -0.3866    0.3263   -0.0062    0.3090   -0.0164   -0.0664    0.2175   -0.2217
    0.0428   -0.1296   -0.0311    0.0256   -0.5486    0.1365    0.3844    0.1620    0.0519   -0.4203    0.0299
   -0.0303   -0.2020    0.2537    0.1262   -0.1707    0.0498   -0.0711    0.3350   -0.1156    0.3451    0.3515
    0.1376    0.0813   -0.1103   -0.3486   -0.0949    0.1048   -0.2727    0.5122    0.3295    0.3951   -0.0785
   -0.0104    0.4138    0.2980   -0.2143   -0.0936    0.4982    0.3858   -0.0315    0.2035   -0.0319   -0.0424
   -0.5539    0.0424   -0.1871   -0.1979   -0.2711    0.0447    0.0022   -0.0011    0.0972    0.0406   -0.0443
    0.4890    0.0267    0.0469   -0.2019   -0.0173   -0.0328   -0.1045   -0.2790    0.2472   -0.0639    0.5019
    0.0702    0.0307    0.4467   -0.4118    0.0052   -0.0809   -0.2038   -0.3469   -0.1030   -0.1087   -0.2257
    0.1622   -0.5495   -0.3357   -0.0994    0.2219    0.4932   -0.0387   -0.0478    0.2024   -0.1963    0.0370
   -0.0789    0.1388    0.1397   -0.0690   -0.1184   -0.0374   -0.4527    0.1117    0.3931   -0.2885   -0.0572
   -0.1649   -0.4186    0.0930   -0.3442   -0.3332   -0.0151    0.0864   -0.3584   -0.0726    0.3936    0.1208
    0.5681    0.0032   -0.0011    0.0510   -0.3618    0.0909   -0.0391    0.1092   -0.3575    0.1174   -0.4097

  Columns 12 through 16

    0.1939   -0.1277   -0.4816   -0.0681   -0.0490
    0.1270   -0.2783    0.3358   -0.5861   -0.0035
   -0.0617    0.1529   -0.0305   -0.0445    0.0018
   -0.3809    0.0289   -0.0373    0.0796   -0.0461
   -0.1595   -0.3285   -0.3012   -0.1756    0.3871
   -0.1891   -0.1258   -0.4306   -0.2165   -0.1488
   -0.3715   -0.4480    0.2436   -0.0520    0.2690
   -0.0126    0.1457   -0.1083   -0.1183   -0.4076
    0.0477    0.1075    0.4550   -0.0907    0.1179
    0.2945   -0.4040    0.0135    0.5232   -0.0692
    0.4091   -0.3656   -0.0691   -0.0622   -0.0203
   -0.4752   -0.1862   -0.0505    0.1799   -0.2964
   -0.2357    0.0567    0.1726    0.2878    0.0263
   -0.0859    0.1501   -0.2201    0.0704    0.6270
    0.1014    0.4094   -0.1107   -0.1278    0.2049
    0.2063   -0.0026    0.0283    0.3563    0.2056
```


Original Image
KLT Spectrum (Eigenvalues)
Reconstructed Image
Difference Between Original and Reconstructe



PROGRAM 20: Apply a mask on image and apply KLT & SVD and compare their results.

```
I_2D = imread('download.jpg');
if size(I_2D, 3) == 3
   I_2D = rgb2gray(I_2D);
end

I_2D = double(I_2D);
[m, n] = size(I_2D);
new_size = min(m, n);
I_2D = imresize(I_2D, [new_size, new_size]);

mask = zeros(new_size, new_size);
mask(1:50, 1:50) = 1;

C_2D = cov(I_2D);
[E_2D, D_2D] = eig(C_2D);

I_2D_KLT = E_2D' * I_2D * E_2D;
I_2D_KLT_Masked = I_2D_KLT .* mask;
I_2D_KLT_Reconstructed = E_2D * I_2D_KLT_Masked * E_2D';

[U, S, V] = svd(I_2D);
S_Masked = S .* mask;
I_2D_SVD_Reconstructed = U * S_Masked * V';

figure;
subplot(3,3,1); imshow(I_2D, []); title('Original Image');
subplot(3,3,2); imshow(log(1 + abs(D_2D)), []); title('KLT Spectrum');
subplot(3,3,3); imshow(log(1 + abs(S)), []); title('SVD Spectrum');

subplot(3,3,4); imshow(I_2D_KLT, []); title('KLT Transformed Image');
subplot(3,3,5); imshow(I_2D_SVD_Reconstructed, []); title('SVD Reconstructed Image');
subplot(3,3,6); imshow(I_2D_KLT_Reconstructed, []); title('KLT Reconstructed Image');

subplot(3,3,7); imshow(abs(I_2D - I_2D_SVD_Reconstructed), []); title('SVD Error');
subplot(3,3,8); imshow(abs(I_2D - I_2D_KLT_Reconstructed), []); title('KLT Error');
subplot(3,3,9); imshow(abs(I_2D_SVD_Reconstructed - I_2D_KLT_Reconstructed), []);
title('Difference SVD vs KLT');

I_2D = imread('download.jpg');
if size(I_2D, 3) == 3
   I_2D = rgb2gray(I_2D);
end

I_2D = double(I_2D);
[m, n] = size(I_2D);
new_size = min(m, n);
I_2D = imresize(I_2D, [new_size, new_size]);
```

```
mask = zeros(new_size, new_size);
mask(1:50, 1:50) = 1;

C_2D = cov(I_2D);
[E_2D, D_2D] = eig(C_2D);

I_2D_KLT = E_2D' * I_2D * E_2D;
I_2D_KLT_Masked = I_2D_KLT .* mask;
I_2D_KLT_Reconstructed = E_2D * I_2D_KLT_Masked * E_2D';

[U, S, V] = svd(I_2D);
S_Masked = S .* mask;
I_2D_SVD_Reconstructed = U * S_Masked * V';

figure;
subplot(3,3,1); imshow(I_2D, []); title('Original Image');
subplot(3,3,2); imshow(log(1 + abs(D_2D)), []); title('KLT Spectrum');
subplot(3,3,3); imshow(log(1 + abs(S)), []); title('SVD Spectrum');

subplot(3,3,4); imshow(I_2D_KLT, []); title('KLT Transformed Image');
subplot(3,3,5); imshow(I_2D_SVD_Reconstructed, []); title('SVD Reconstructed Image');
subplot(3,3,6); imshow(I_2D_KLT_Reconstructed, []); title('KLT Reconstructed Image');

subplot(3,3,7); imshow(abs(I_2D - I_2D_SVD_Reconstructed), []); title('SVD Error');
subplot(3,3,8); imshow(abs(I_2D - I_2D_KLT_Reconstructed), []); title('KLT Error');
subplot(3,3,9); imshow(abs(I_2D_SVD_Reconstructed - I_2D_KLT_Reconstructed), []);
title('Difference SVD vs KLT');
```

OUTPUT:

Figure 1

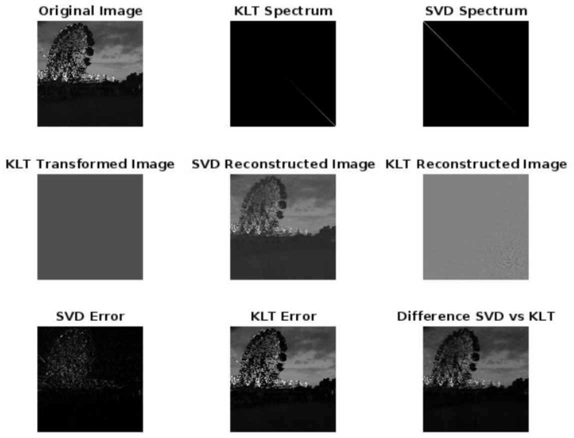
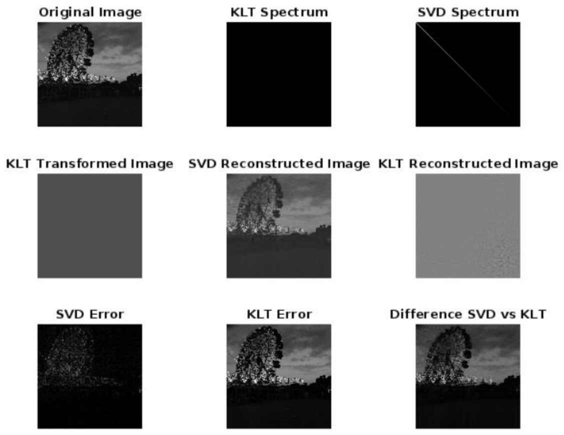


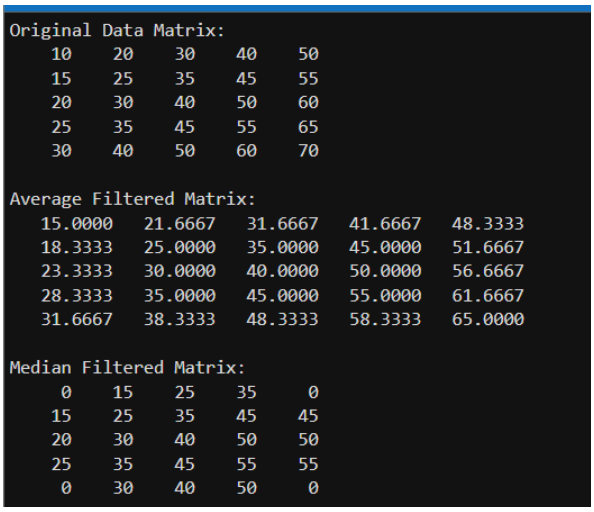
Figure 2



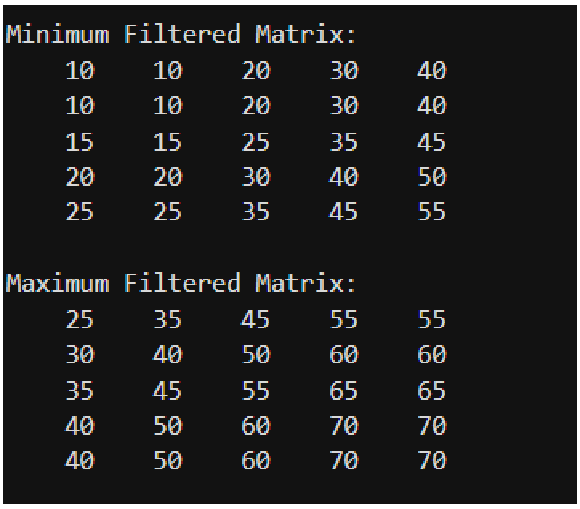
PROGRAM 21: Apply Average, Median, Min, and Max Filters & Comparing Metrics.

```
clc; clear; close all;
data = [
   10 20 30 40 50;
   15 25 35 45 55;
   20 30 40 50 60;
   25 35 45 55 65;
   30 40 50 60 70
];
avg_filter = fspecial('average', [3 3]);
filtered_avg = imfilter(data, avg_filter, 'replicate');
filtered_median = medfilt2(data, [3 3]);
filtered_min = ordfilt2(data, 1, ones(3,3), 'symmetric');
filtered_max = ordfilt2(data, 9, ones(3,3), 'symmetric');
disp('Original Data Matrix:');
disp(data);
disp('Average Filtered Matrix:');
disp(filtered_avg);
disp('Median Filtered Matrix:');
disp(filtered_median);
disp('Minimum Filtered Matrix:');
disp(filtered_min);
disp('Maximum Filtered Matrix:');
disp(filtered_max);
```

Output:



```
Original Data Matrix:
    10    20    30    40    50
    15    25    35    45    55
    20    30    40    50    60
    25    35    45    55    65
    30    40    50    60    70

Average Filtered Matrix:
   15.0000   21.6667   31.6667   41.6667   48.3333
   18.3333   25.0000   35.0000   45.0000   51.6667
   23.3333   30.0000   40.0000   50.0000   56.6667
   28.3333   35.0000   45.0000   55.0000   61.6667
   31.6667   38.3333   48.3333   58.3333   65.0000

Median Filtered Matrix:
     0    15    25    35     0
    15    25    35    45    45
    20    30    40    50    50
    25    35    45    55    55
     0    30    40    50     0
```

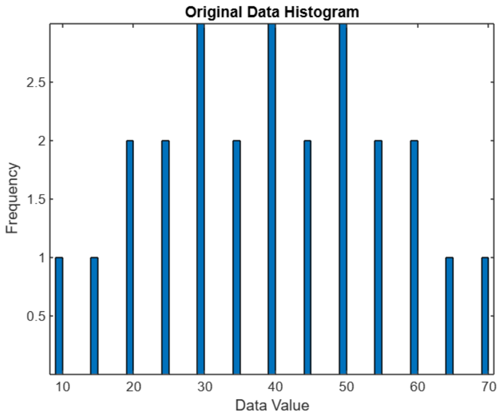


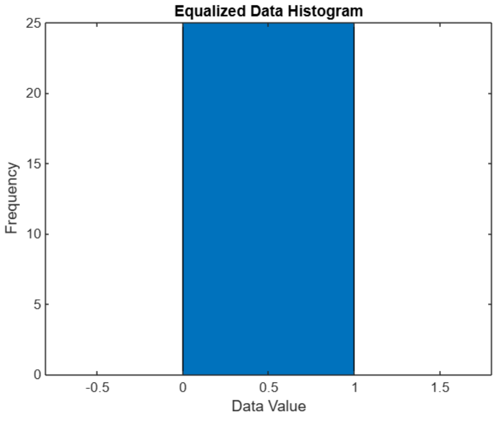
```
Minimum Filtered Matrix:
    10    10    20    30    40
    10    10    20    30    40
    15    15    25    35    45
    20    20    30    40    50
    25    25    35    45    55

Maximum Filtered Matrix:
    25    35    45    55    55
    30    40    50    60    60
    35    45    55    65    65
    40    50    60    70    70
    40    50    60    70    70
```

PROGRAM 22: Apply Gamma, Log, Square, and Square Root Transformations.

```
 clc; clear; close all;
data = [
   10 20 30 40 50;
   15 25 35 45 55;
   20 30 40 50 60;
   25 35 45 55 65;
   30 40 50 60 70
];
data_vector = data(:);
[counts, bin_edges] = histcounts(data_vector, 'BinMethod', 'integer');
figure;
bar(bin_edges(1:end-1), counts, 'BarWidth', 1);
title('Original Data Histogram');
xlabel('Data Value');
ylabel('Frequency');
data_eq = histeq(data, numel(unique(data_vector)));
data_eq_vector = data_eq(:);
[counts_eq, bin_edges_eq] = histcounts(data_eq_vector, 'BinMethod', 'integer');
figure;
bar(bin_edges_eq(1:end-1), counts_eq, 'BarWidth', 1);
title('Equalized Data Histogram');
xlabel('Data Value');
ylabel('Frequency');
disp('Original Data Matrix:');
disp(data);
disp('Equalized Data Matrix:');
disp(data_eq);
```

Output:

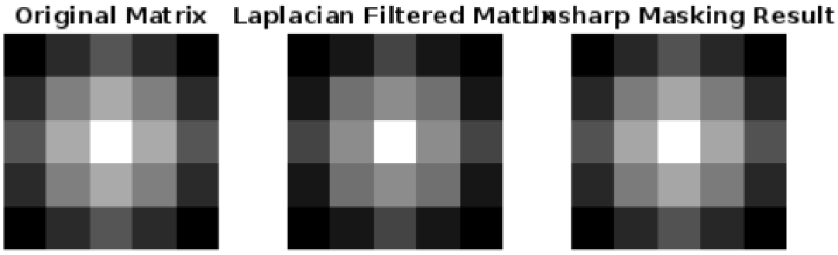




PROGRAM 23: Apply local operations or filters to a given image and compare the results based on the output image.

```
clc; clear; close all;
img = [
    0.2, 0.3, 0.4, 0.3, 0.2;
    0.3, 0.5, 0.6, 0.5, 0.3;
    0.4, 0.6, 0.8, 0.6, 0.4;
    0.3, 0.5, 0.6, 0.5, 0.3;
    0.2, 0.3, 0.4, 0.3, 0.2
];
laplacian_filter = fspecial('laplacian', 0.2);
sharpened = img - imfilter(img, laplacian_filter, 'replicate');
gaussian_blur = imgaussfilt(img, 2);
unsharp_mask = img + 1.5 * (img - gaussian_blur);
figure;
subplot(1,3,1), imshow(img, []), title('Original Matrix');
subplot(1,3,2), imshow(sharpened, []), title('Laplacian Filtered Matrix');
subplot(1,3,3), imshow(unsharp_mask, []), title('Unsharp Masking Result');
```

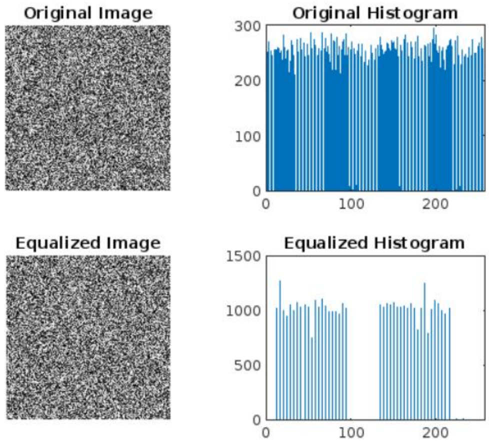
Output:

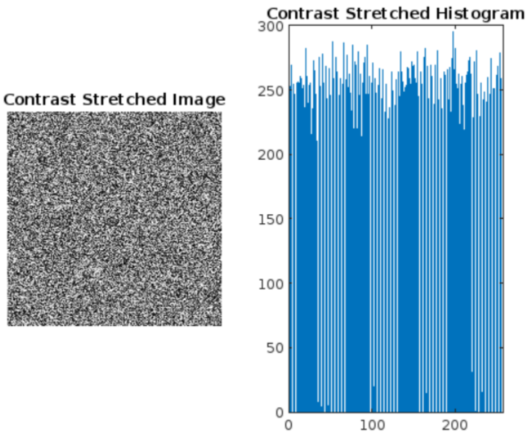


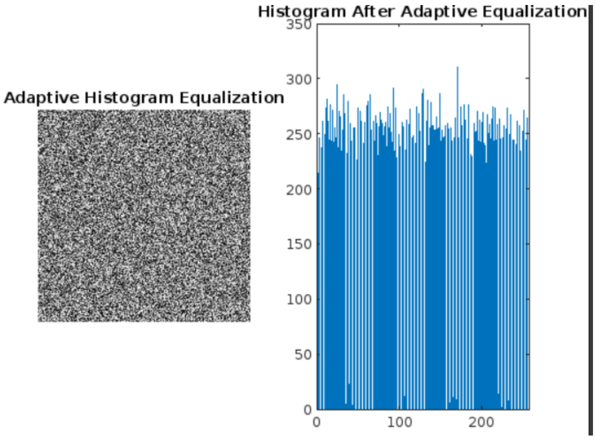
PROGRAM 24: Calculate histogram of a given image and apply the various histogram enhanced methods and compare the result based on the resultant image and histograms. use matrix

```
imageMatrix = uint8(rand(256) * 255);
figure;
subplot(2, 2, 1); imshow(imageMatrix); title('Original Image');
subplot(2, 2, 2); bar(imhist(imageMatrix)); title('Original Histogram');
imageEqualized = histeq(imageMatrix);
subplot(2, 2, 3); imshow(imageEqualized); title('Equalized Image');
subplot(2, 2, 4); bar(imhist(imageEqualized)); title('Equalized Histogram');
min_val = double(min(imageMatrix(:)));
max_val = double(max(imageMatrix(:)));
contrastStretched = uint8(255 * (double(imageMatrix) - min_val) / (max_val - min_val));
figure;
subplot(1, 2, 1); imshow(contrastStretched); title('Contrast Stretched Image');
subplot(1, 2, 2); bar(imhist(contrastStretched)); title('Contrast Stretched Histogram');
imageCLAHE = adapthisteq(imageMatrix);
figure;
subplot(1, 2, 1); imshow(imageCLAHE); title('Adaptive Histogram Equalization');
subplot(1, 2, 2); bar(imhist(imageCLAHE)); title('Histogram After Adaptive Equalization');
```

Output:


Original Image
Original Histogram
Equalized Image
Equalized Histogram


Contrast Stretched Image
Contrast Stretched Histogram


Adaptive Histogram Equalization
Histogram After Adaptive Equalization