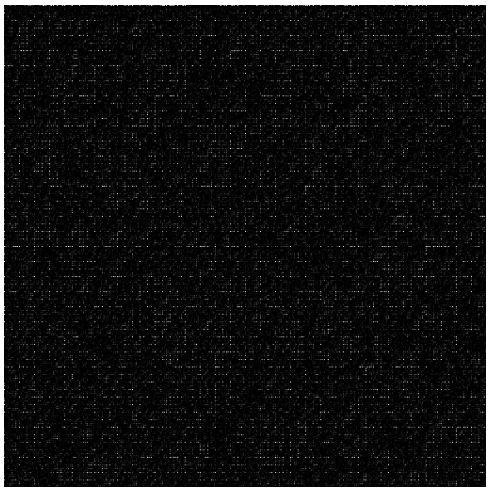
```
********************************************************************************
Aufgabe 2.1
********************************************************************************

> b1 = [128, 64; 16, 32]
b1 =
   128    64
    16    32

> nb1 = normalizeAuto(b1)
nb1 =

   1.00000   0.00000
  -0.75000  -0.50000

> hb1 = h2 * nb1
hb1 =

   0.17678  -0.35355
   1.23744   0.35355

> hb1h = hb1 * h2
hb1h =

  -0.12500   0.37500
   1.12500   0.62500

> hhb1h = h2 * hb1h
hhb1h =

   0.70711   0.70711
  -0.88388  -0.17678

> hhb1hh = hhb1h * h2
hhb1hh =

   1.0000e+00   7.8496e-17
  -7.5000e-01  -5.0000e-01

> denormalize(hhb1hh, 128)
ans =

  255  128
   32   64
```

```
********************************************************************************
Aufgabe 2.2
********************************************************************************

> b2 = [17, 205; 205, 17]
b2 =

    17   205
   205    17

> nb2 = normalizeAuto (b2)
nb2 =

  -0.83415   1.00000
   1.00000  -0.83415

> hb2 = h2 * nb2
hb2 =

   0.11728   0.11728
  -1.29694   1.29694

> hb2h = hb2 * h2
hb2h =

   0.16585   0.00000
   0.00000  -1.83415

> hhb2h = h2 * hb2h
hhb2h =

   0.11728  -1.29694
   0.11728   1.29694

> hhb2hh = hhb2h * h2
hhb2hh =

  -0.83415   1.00000
   1.00000  -0.83415

> denormalize (hhb2hh, 205)
ans =

   21  255
  255   21
```

```
********************************************************************************
Aufgabe 2.3
********************************************************************************

> b3 = [0, 204, 102, 204; 204, 255, 51, 51; 204, 102, 153, 51; 153, 51, 51, 0]
b3 =

     0   204   102   204
   204   255    51    51
   204   102   153    51
   153    51    51     0

> nb3 = normalizeAuto (b3)
nb3 =

  -1.00000   0.60000  -0.20000   0.60000
   0.60000   1.00000  -0.60000  -0.60000
   0.60000  -0.20000   0.20000  -0.60000
   0.20000  -0.60000  -0.60000  -1.00000

> hb3 = h4 * nb3
hb3 =

   2.0000e-01   4.0000e-01  -6.0000e-01  -8.0000e-01
  -6.0000e-01   5.5484e-17   6.0000e-01   8.0000e-01
  -6.0000e-01   1.2000e+00  -2.0000e-01   8.0000e-01
  -1.0000e+00  -4.0000e-01  -2.0000e-01   4.0000e-01

> hb3h = hb3 * h4
hb3h =

  -4.0000e-01  -5.5511e-17   1.0000e+00  -2.0000e-01
   4.0000e-01  -4.0000e-01  -1.0000e+00  -2.0000e-01
   6.0000e-01  -1.4000e+00  -2.7756e-17  -4.0000e-01
  -6.0000e-01  -6.0000e-01  -8.0000e-01  -1.1101e-16

> hhb3h = h4 * hb3h
hhb3h =

   8.3267e-17  -1.2000e+00  -4.0000e-01  -4.0000e-01
   2.0000e-01  -2.0000e-01   1.4000e+00  -2.0000e-01
   8.3267e-17   8.0000e-01   4.0000e-01   9.7138e-17
  -1.0000e+00   6.0000e-01   6.0000e-01   2.0000e-01

> hhb3hh = hhb3h * h4
hhb3hh =

  -1.00000   0.60000  -0.20000   0.60000
   0.60000   1.00000  -0.60000  -0.60000
   0.60000  -0.20000   0.20000  -0.60000
   0.20000  -0.60000  -0.60000  -1.00000

> denormalize (hhb3hh, 255)
ans =

    0  204  102  204
  204  255   51   51
  204  102  153   51
  153   51   51    0
```

```
#*******************************************************************************
#Aufgabe 3
#*******************************************************************************
source ub1.m;

# Lesen des Bilds
#*******************************************************************************
B = imread("lena-bw.png");

# I) normalisieren, transformieren mit Hadamard, rausschreiben nach "Bfreq.png"
#*******************************************************************************
B_norm = normalizePicture(B);

B_freq = h512 * B_norm * h512;

imwrite(B_freq, "Bfreq.png")


# II) rücktransformieren, auf den Bereich 0..255 skalieren, schrebien nach
"Bback.png"
#*******************************************************************************
B_back = h512 * B_freq * h512;

B_back_den = denormalizeToPicture(B_back);

imwrite(B_back_den, "Bback.png")

# Das rücktransformierte Bild gleicht dem Original!



# spaßershalber ein Tiefpassfilter nach "Blow.png"
#*******************************************************************************
B_low = h512 * lowpass(B_freq, 0) * h512;

B_low_den = denormalizeToPicture(B_low);

imwrite(B_low_den, "Blow.png")
```

Bfreq.png



Bback.png

Blow.png

```
************************************************************************
Definitionen, die im Vorhergehenden benutzt wurden
************************************************************************

h2 = 1 / sqrt(2) * [1,1;1,-1];
h4 = 1 / sqrt(2) * [h2, h2; h2, -h2];
h8 = 1 / sqrt(2) * [h4, h4; h4, -h4];
h16 = 1 / sqrt(2) * [h8, h8; h8, -h8];
h32 = 1 / sqrt(2) * [h16, h16; h16, -h16];
h64 = 1 / sqrt(2) * [h32, h32; h32, -h32];
h128 = 1 / sqrt(2) * [h64, h64; h64, -h64];
h256 = 1 / sqrt(2) * [h128, h128; h128, -h128];
h512 = 1 / sqrt(2) * [h256, h256; h256, -h256];

b1 = [128, 64; 16, 32] ;
b2 = [17, 205; 205, 17];
b3 = [0, 204, 102, 204; 204, 255, 51, 51; 204, 102, 153, 51; 153, 51, 51, 0];

# nomalisiere Matrix x mit dem größten vorkommenden Wert
function result = normalizeAuto(x)
      maximum = max(max(x));
      result = double(x) / maximum * 2 - 1;
end;

# denomalisiere Matrix x mit dem Wert factor (Rückskalieren auf ursprünglichen
Wertebereich)
function result = denormalize(x, factor)
      result = (x + 1) / 2.0 * factor;
end;

function result = normalizePicture(x)
      result = double(x) / 255.0 * 2 - 1;
end;

function result = denormalizeToPicture(x)
      result = uint8((x + 1) / 2.0 * 255);
end;




# Werte einer Matrix auf den Bereich [0..1] skalieren
function result = normalize01(x)
      minval = min(min(x))
      maxval = max(max(x))
      x = x - minval;
      x = x / (maxval - minval)
      result = x;
end;

function result = lowpass(x, threshold)
      y = x;
      y(find (y > threshold)) = 0;
      result = y;
end;
```