JPEG using DCT

NAMES & IDs :

Ahmed Mahmoud Mohamed ElDokmak

18010248

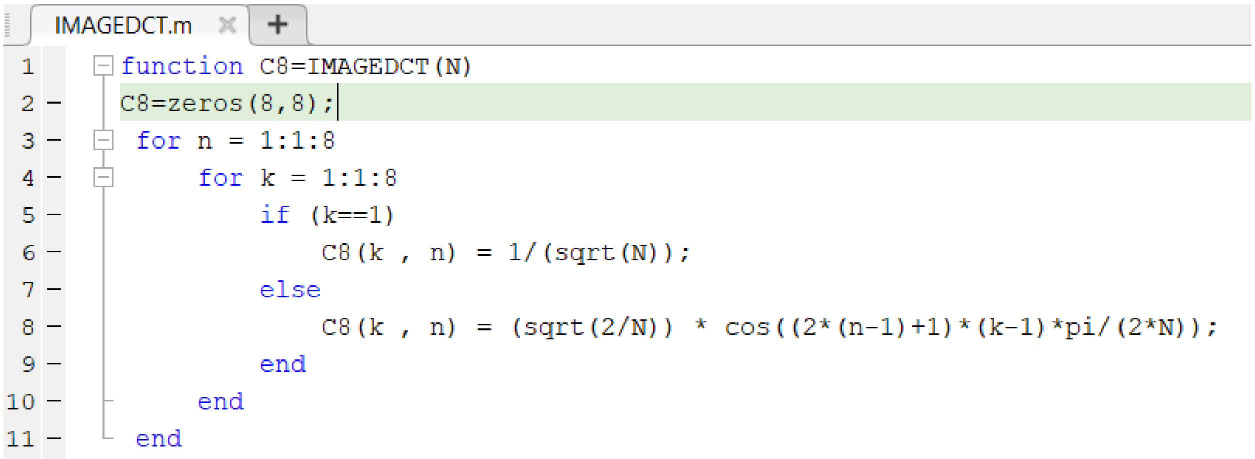
Basel Moustafa Sielem Soliman

18010459

Youssouf Muhammad Youssouf Alghmarawy

18012142

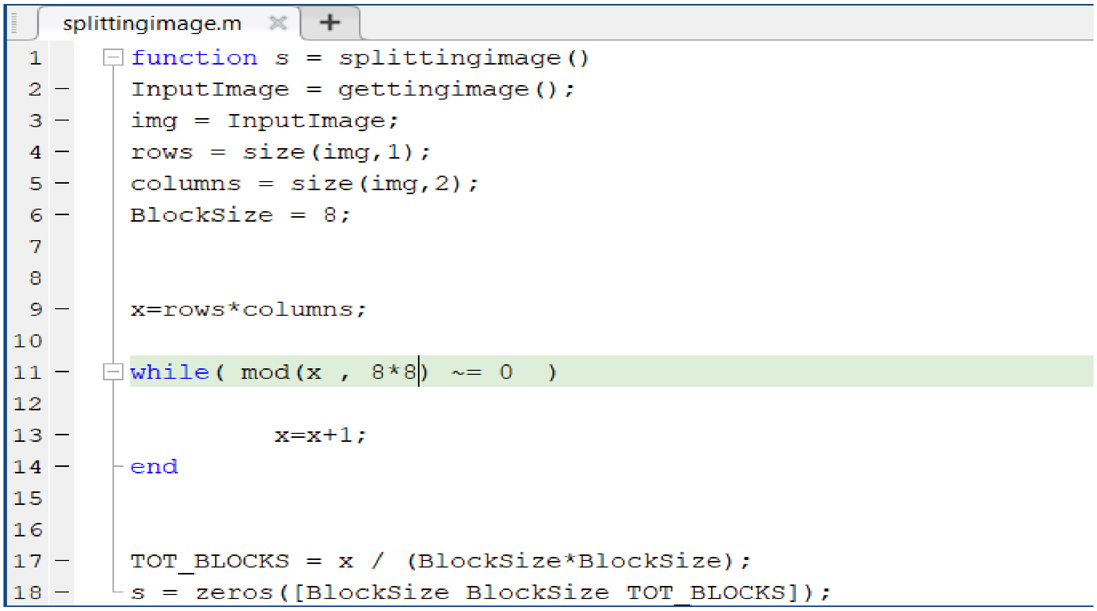
1- Write a script that computes C8 and store the result as a variable named C8 in a file with the same name



```
function C8=IMAGEDCT(N)
C8=zeros(8,8);
 for n = 1:1:8
     for k = 1:1:8
         if (k==1)
             C8(k , n) = 1/(sqrt(N));
         else
             C8(k , n) = (sqrt(2/N)) * cos((2*(n-1)+1)*(k-1)*pi/(2*N));
         end
     end
 end
```

In this function we compute DCT for C8 by making C8 = 1/(sqrt(N))in case k = 0 and = (sqrt(2/N)) * cos((2*(n-1)+1)*(k-1)*pi/(2*N))in case of k is between 1 and 7.

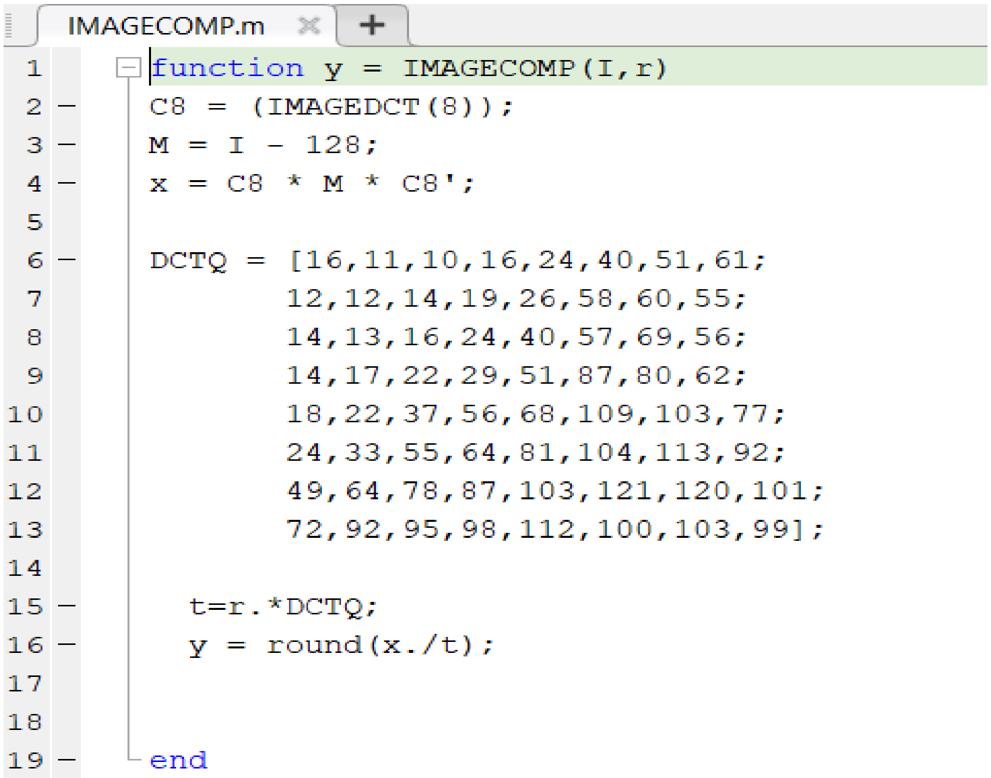
2- Write a function that will split an image matrix into small blocks of size 8*8



```
function s = splittingimage()
InputImage = gettingimage();
img = InputImage;
rows = size(img,1);
columns = size(img,2);
BlockSize = 8;


x=rows*columns;

while( mod(x , 8*8)  ~= 0   )

         x=x+1;
end


TOT_BLOCKS = x /  (BlockSize*BlockSize);
s = zeros([BlockSize BlockSize TOT_BLOCKS]);
```

Here we get image from function getting image and calculate rows and columns of the image to calculate its area and check if area is divided by 8*8 if not we add 1 to area and check again and repeat to make aria dividable by 8*8 then we can calculate the total number of blocks so we can split image into small blocks 8*8.

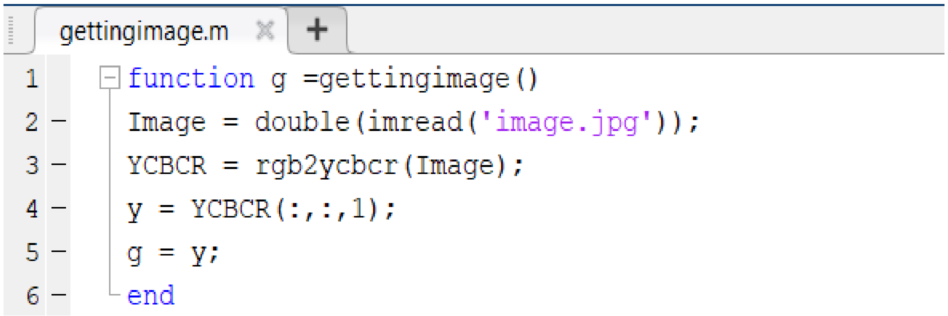
3- Write a function that takes as argument a scaling factor r, the quantization matrix DCTQ and a block split DCT of an image to produce the quantized block split image.



```
function y = IMAGECOMP(I,r)
C8 = (IMAGEDCT(8));
M = I - 128;
x = C8 * M * C8';

DCTQ = [16,11,10,16,24,40,51,61;
       12,12,14,19,26,58,60,55;
       14,13,16,24,40,57,69,56;
       14,17,22,29,51,87,80,62;
       18,22,37,56,68,109,103,77;
       24,33,55,64,81,104,113,92;
       49,64,78,87,103,121,120,101;
       72,92,95,98,112,100,103,99];

  t=r.*DCTQ;
  y = round(x./t);


end
```

Here we need to compress image by quantization so we call C8 and image small blocks 8*8 from main and factor that user inputs then level off this block by subtracting 128 from each entry the calculate x as shown then we calculate t to get compressed image.

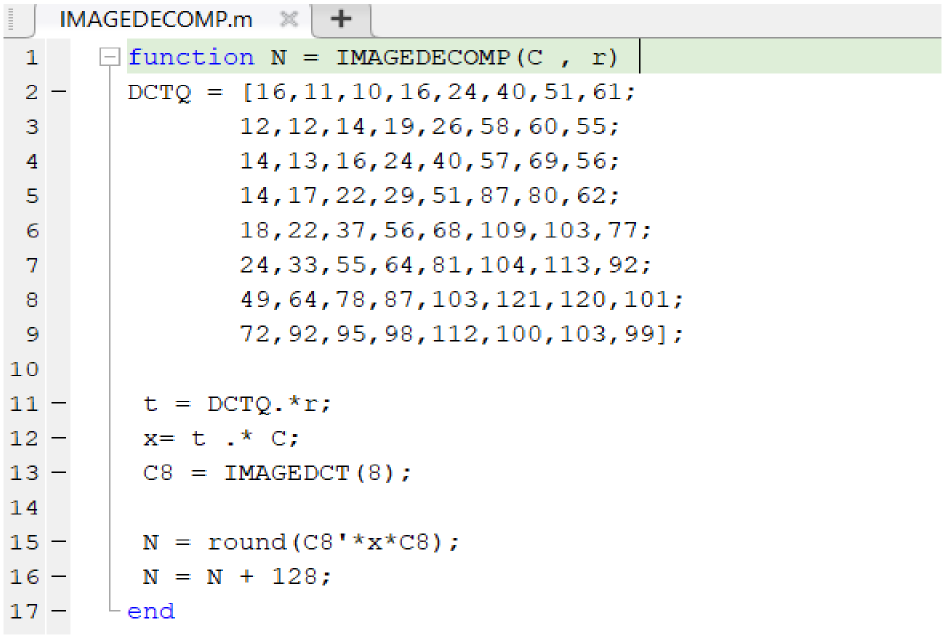
4- Write a function to do the rescaling Do not forget to convert the input signal to double, as r is double. The output is double.



```
function g =gettingimage()
Image = double(imread('image.jpg'));
YCBCR = rgb2ycbcr(Image);
y = YCBCR(:,:,1);
g = y;
end
```

In this function we get image usnig instruction imread and convert it to double to be used in dct and then convert to ycbcr these color spaces are suitble for dct.

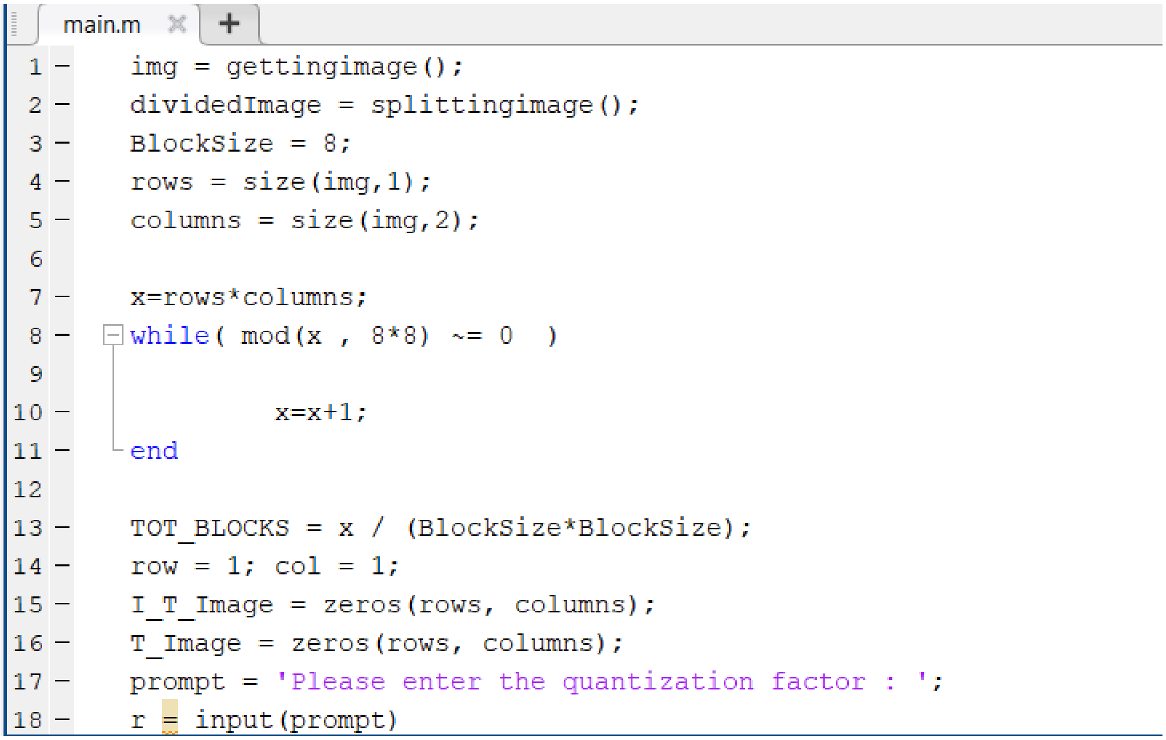
5- Write a line of code that computes the inverse block DCT using the routine "blockDCT".



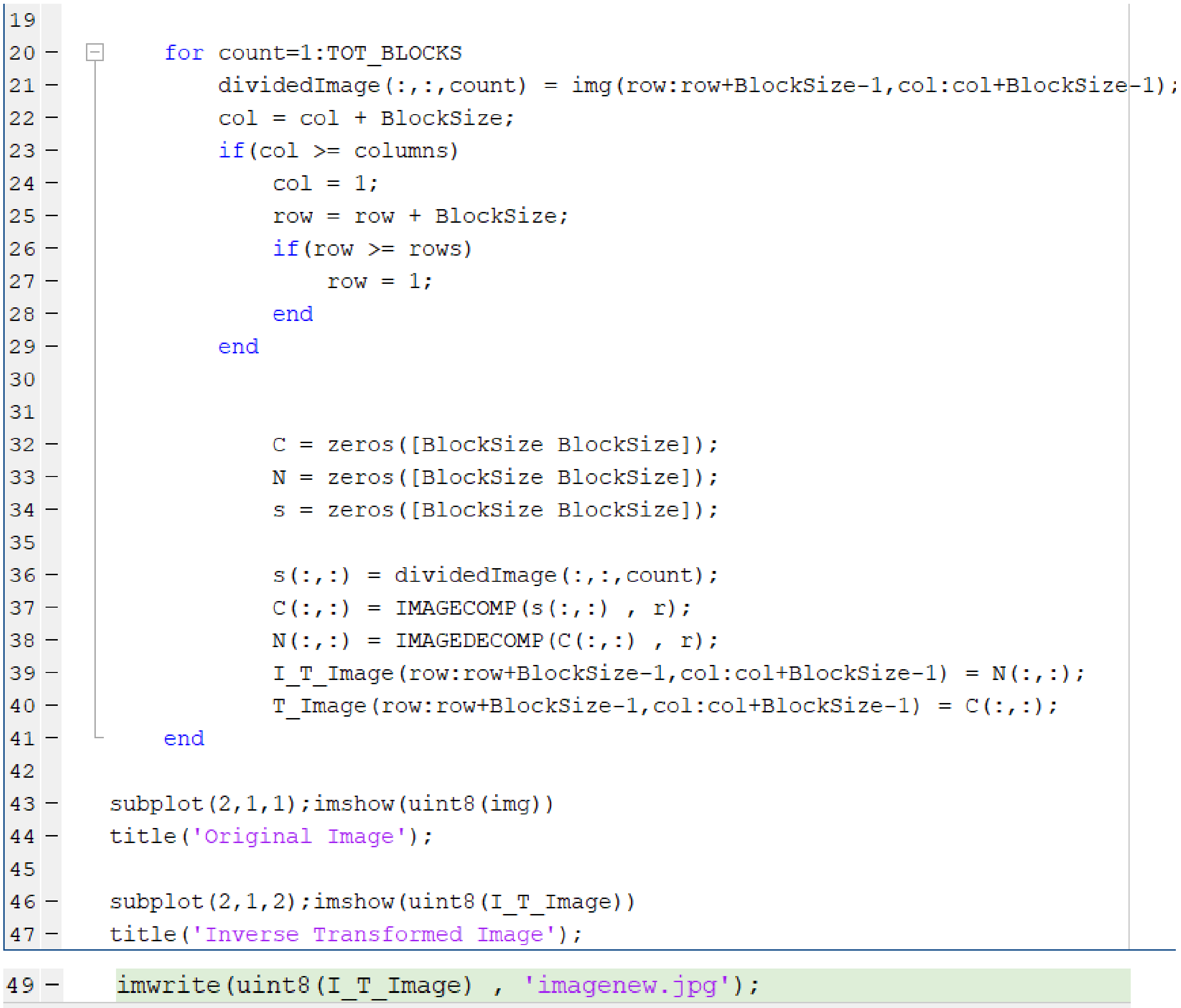
```
function N = IMAGEDECOMP(C , r)
DCTQ = [16,11,10,16,24,40,51,61;
        12,12,14,19,26,58,60,55;
        14,13,16,24,40,57,69,56;
        14,17,22,29,51,87,80,62;
        18,22,37,56,68,109,103,77;
        24,33,55,64,81,104,113,92;
        49,64,78,87,103,121,120,101;
        72,92,95,98,112,100,103,99];

 t = DCTQ.*r;
 x= t .* C;
 C8 = IMAGEDCT(8);

 N = round(C8'*x*C8);
 N = N + 128;
end
```

This function is used to decmopress image again so we call compressed one from main and factor that user had input then calculate t and x as shown the we call C8 again to ger N then add 128 to bring image again in range 0 to 255.

6- Merging the reconstructed split image into a conventional image. write a function that recombines the blocks. The output must be uint8.



```
img = gettingimage();
dividedImage = splittingimage();
BlockSize = 8;
rows = size(img,1);
columns = size(img,2);

x=rows*columns;
while( mod(x , 8*8) ~= 0  )

         x=x+1;
end

TOT_BLOCKS = x / (BlockSize*BlockSize);
row = 1; col = 1;
I_T_Image = zeros(rows, columns);
T_Image = zeros(rows, columns);
prompt = 'Please enter the quantization factor : ';
r = input(prompt)
```



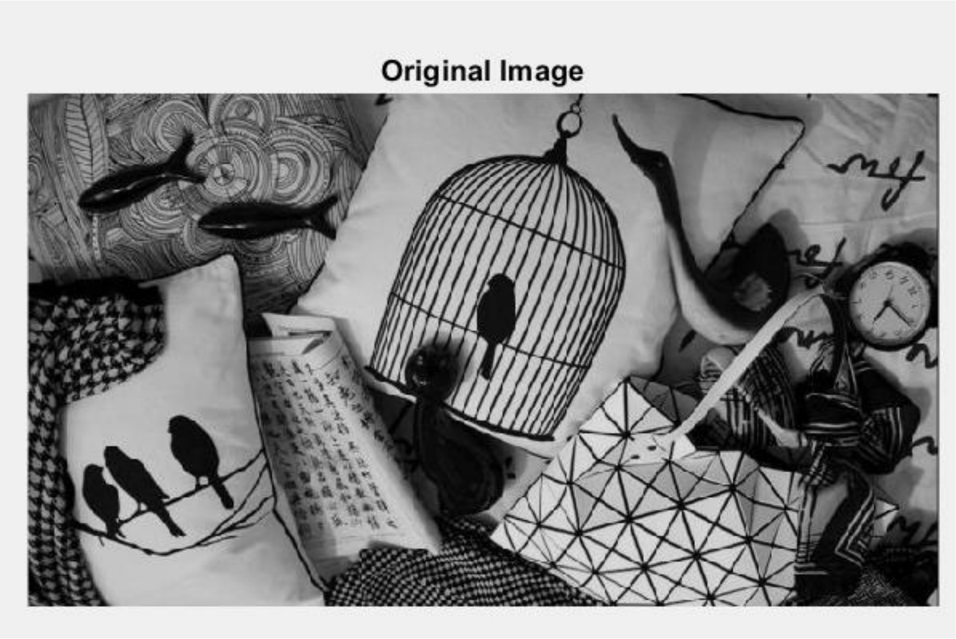
```

    for count=1:TOT_BLOCKS
        dividedImage(:,:,count) = img(row:row+BlockSize-1,col:col+BlockSize-1);
        col = col + BlockSize;
        if(col >= columns)
            col = 1;
            row = row + BlockSize;
            if(row >= rows)
                row = 1;
            end
        end


            C = zeros([BlockSize BlockSize]);
            N = zeros([BlockSize BlockSize]);
            s = zeros([BlockSize BlockSize]);

            s(:,:) = dividedImage(:,:,count);
            C(:,:) = IMAGECOMP(s(:,:) , r);
            N(:,:) = IMAGEDECOMP(C(:,:) , r);
            I_T_Image(row:row+BlockSize-1,col:col+BlockSize-1) = N(:,:);
            T_Image(row:row+BlockSize-1,col:col+BlockSize-1) = C(:,:);
    end

subplot(2,1,1);imshow(uint8(img))
title('Original Image');

subplot(2,1,2);imshow(uint8(I_T_Image))
title('Inverse Transformed Image');
```

```
imwrite(uint8(I_T_Image) , 'imagenew.jpg');
```

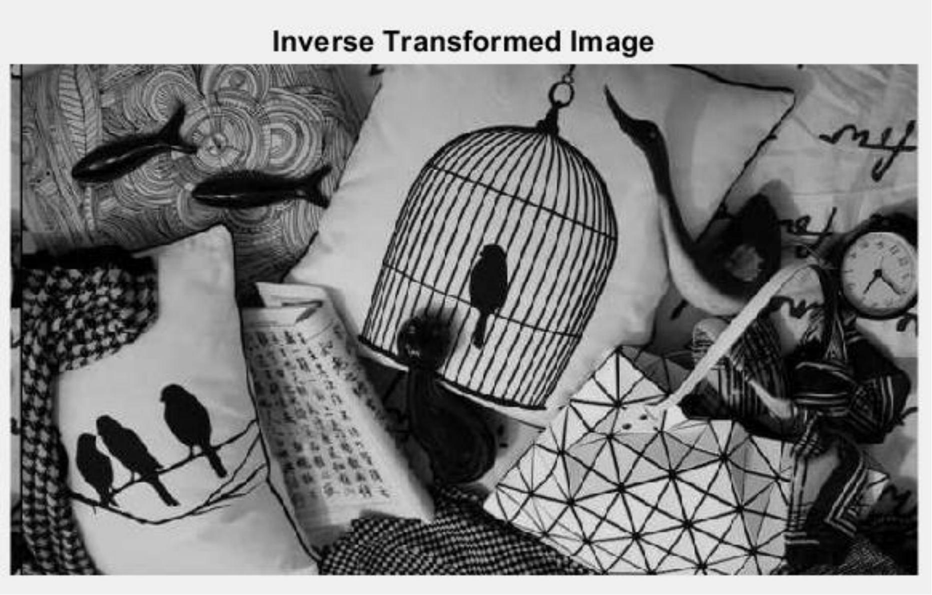
Here is the main file first we get image using gettingimage used above then divided image from splitting image function then we call splittingimage then calculate totalblocks again as previous and ask the user to input quantization factor (r) then we enter for loop to start filling these blocks then we give these image blocks and factor (r) to IMGCOMP function and this output and factor (r) goes to DECCOMP function again get back decompressed image then we show out the original image and inverse transformed image.

7- Run

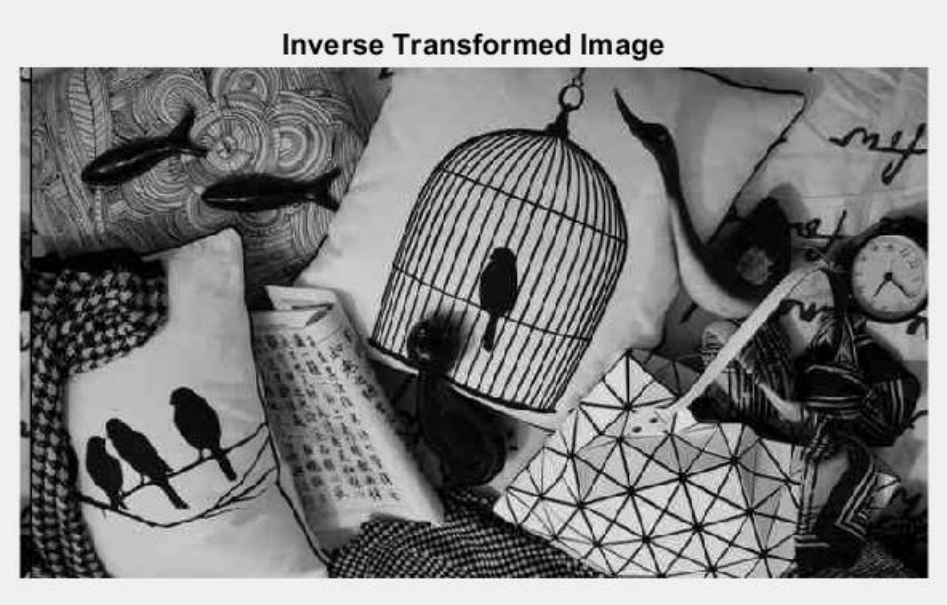
Original image ,size = 65KB



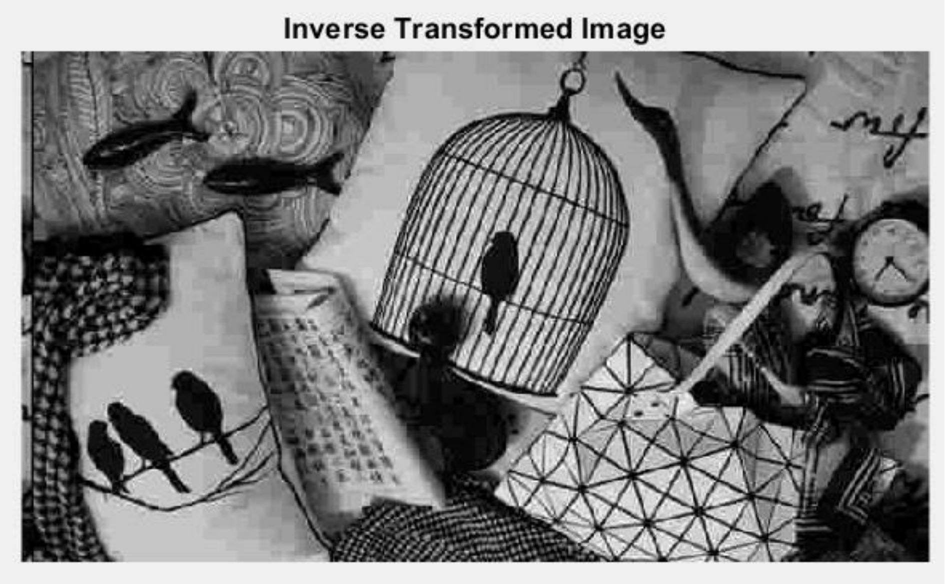
Output image with factor = 1 ,size = 51KB



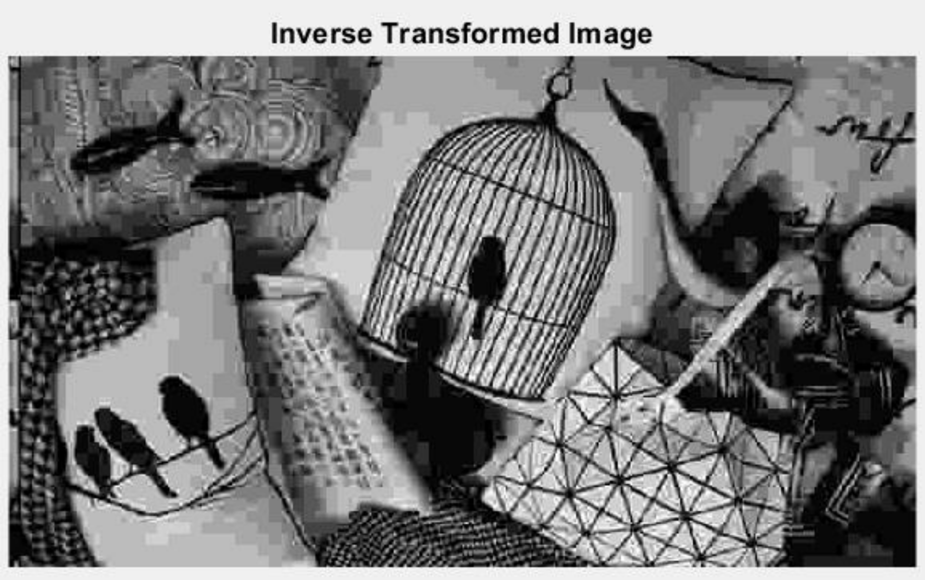
Output image with factor = 2   , size = 48KB



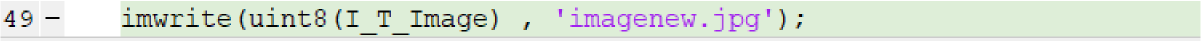
Output image with factor = 5   ,size = 33KB



Output image with factor = 10 ,size = 25KB



From previos photos its obvious that the quality of the photo decreases as quantization factor increases and if we check size of output image using following instruction



```
imwrite(uint8(I_T_Image) , 'imagenew.jpg');
```

we will find that it decreases too as illustrated above each photo of the previous ones.