```
function [sim, compratio, B, B2, compimg] = DCTCOMP(img,quality)
%  Takes in an image matrix and quality value from 0 to 15, 0
%  creates an all black image while 15 simply recreates the original.
%  Outputs similarity, compression ratio, transform of original image,
masked transformed image and the compressed image respectively

%% Process in Image

%  Create a matrix for each layer of the image
img1 = img(:,:,1);
img2 = img(:,:,2);
img3 = img(:,:,3);

%% Discrete Cosine Transform (DCT)

%  Create an 8x8 DCT matrix with dctmtx
T = dctmtx(8);

%  Define DCT function to pass to blockproc
dct = @(block_struct) T * block_struct.data * T';

%  Perform DCT on each layer with blockproc and the PadPartialBlocks
command set
%  to true in order to make up for the fact that the picture's
dimensions
%  might not be multiples of 8
B1 = blockproc(img1,[8 8],dct, 'PadPartialBlocks',true);
B2 = blockproc(img2,[8 8],dct, 'PadPartialBlocks',true);
B3 = blockproc(img3,[8 8],dct, 'PadPartialBlocks',true);

%  Compile DCT of image into matrix B
B(:,:,1)=B1;
B(:,:,2)=B2;
B(:,:,3)=B3;

%% Compression

%  Define several masking matricies to remove different amounts of
entries of the DCT of the image that
%  don't contain vital visual informaiton based on the indicated
quality
%  level
mask0 = [0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0];

mask1 = [1   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
         0   0   0   0   0   0   0   0
```

```
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];

mask2 = [1    1    0    0    0    0    0    0
          1    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];

mask3 = [1    1    1    0    0    0    0    0
          1    1    0    0    0    0    0    0
          1    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];

mask4 = [1    1    1    1    0    0    0    0
          1    1    1    0    0    0    0    0
          1    1    0    0    0    0    0    0
          1    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];

mask5 = [1    1    1    1    1    0    0    0
          1    1    1    1    0    0    0    0
          1    1    1    0    0    0    0    0
          1    1    0    0    0    0    0    0
          1    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];

mask6 = [1    1    1    1    1    1    0    0
          1    1    1    1    1    0    0    0
          1    1    1    1    0    0    0    0
          1    1    1    0    0    0    0    0
          1    1    0    0    0    0    0    0
          1    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];

mask7 = [1    1    1    1    1    1    1    0
          1    1    1    1    1    1    0    0
          1    1    1    1    1    0    0    0
          1    1    1    1    0    0    0    0
          1    1    1    0    0    0    0    0
          1    1    0    0    0    0    0    0
          1    0    0    0    0    0    0    0
          0    0    0    0    0    0    0    0];
```

```
mask8 = [1   1   1   1   1   1   1   1
         1   1   1   1   1   1   1   0
         1   1   1   1   1   1   0   0
         1   1   1   1   1   0   0   0
         1   1   1   1   0   0   0   0
         1   1   1   0   0   0   0   0
         1   1   0   0   0   0   0   0
         1   0   0   0   0   0   0   0];

mask9 = [1   1   1   1   1   1   1   1
         1   1   1   1   1   1   1   1
         1   1   1   1   1   1   1   0
         1   1   1   1   1   1   0   0
         1   1   1   1   1   0   0   0
         1   1   1   1   0   0   0   0
         1   1   1   0   0   0   0   0
         1   1   0   0   0   0   0   0];

mask10 = [1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   0
          1   1   1   1   1   1   0   0
          1   1   1   1   1   0   0   0
          1   1   1   1   0   0   0   0
          1   1   1   0   0   0   0   0];

mask11 = [1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   0
          1   1   1   1   1   1   0   0
          1   1   1   1   1   0   0   0
          1   1   1   1   0   0   0   0];

mask12 = [1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   0
          1   1   1   1   1   1   0   0
          1   1   1   1   1   0   0   0];

mask13 = [1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   0
          1   1   1   1   1   1   0   0];

mask14 = [1   1   1   1   1   1   1   1
          1   1   1   1   1   1   1   1
```

```
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   0];

mask15 = [1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1
           1   1   1   1   1   1   1   1];

if quality == 0
    mask = mask0;
elseif quality == 1
    mask = mask1;
elseif quality == 2
    mask = mask2;
elseif quality == 3
    mask = mask3;
elseif quality == 4
    mask = mask4;
elseif quality == 5
    mask = mask5;
elseif quality == 6
    mask = mask6;
elseif quality == 7
    mask = mask7;
elseif quality == 8
    mask = mask8;
elseif quality == 9
    mask = mask9;
elseif quality == 10
    mask = mask10;
elseif quality == 11
    mask = mask11;
elseif quality == 12
    mask = mask12;
elseif quality == 13
    mask = mask13;
elseif quality == 14
    mask = mask14;
else
    mask = mask15;
end


%  Apply mask with blockproc to each layer
B21 = blockproc(B1,[8 8],@(block_struct) mask .* block_struct.data,
'PadPartialBlocks',true);
B22 = blockproc(B2,[8 8],@(block_struct) mask .* block_struct.data,
'PadPartialBlocks',true);
B23 = blockproc(B3,[8 8],@(block_struct) mask .* block_struct.data,
```

```
'PadPartialBlocks',true);

%  Compile compressed image matrix into B2
B2(:,:,1)=B21;
B2(:,:,2)=B22;
B2(:,:,3)=B23;




%% Recreate Compressed Image

%  Define the inverse transform function to be passed to blockproc, and
%  performed on each layer of the compressed DCT image
invdct = @(block_struct) T' * block_struct.data * T;
compimg1 = blockproc(B21,[8 8],invdct,'PadPartialBlocks',true);
compimg2 = blockproc(B22,[8 8],invdct,'PadPartialBlocks',true);
compimg3 = blockproc(B23,[8 8],invdct,'PadPartialBlocks',true);

%  Compile compressed image into compimg matrix
clear compimg;
compimg(:,:,1)=compimg1;
compimg(:,:,2)=compimg2;
compimg(:,:,3)=compimg3;


% Use Frobenius norms to compare original and compressed matricies
cn1 = norm((compimg1-img1),'fro');
cn2 = norm((compimg2-img2),'fro');
cn3 = norm((compimg3-img3),'fro');

n1 = norm(img1,'fro');
n2 = norm(img2,'fro');
n3 = norm(img3,'fro');

sim =100*(1-((cn1/n1)+(cn2/n2)+(cn3/n3))/3);

%  Find the size of each version of the image and compare to find the
compression ratio
[w,h,z] = size( B );
img_size = 24*w*h;
pix = 3*w*h;

num = sum(sum(sum(B2(:,:,:) ~= 0)));
tot_size = num*8 + (pix - num);
compratio = img_size/tot_size;
 end
```