```
clc;
clear all;
close all;
I=imread("VK.jpg");
if size(I,3)==3
I=rgb2gray(I);
end
I=double(I);
%Loading the grayscale image and converting it into double for DCT operations.
Q=[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 35 55 64 81 104 113 92;
49 64 78 87 103 121 120 101;
72 92 95 98 112 100 103 99];
%Defining the standard JPEG quantization matrix.
factor=10;
Q=Q*factor;
%Increasing quantization values to introduce stronger compression effects.
blockSize=8;
%JPEG processes the image in fixed 8x8 pixel blocks.
[m,n]=size(I);
reconstructed=zeros(m,n);
%Initializing an empty matrix to store reconstructed compressed output.
for i=1:blockSize:(m-blockSize+1)
for j=1:blockSize:(n-blockSize+1)
block=I(i:i+7,j:j+7);
%Extracting one valid 8x8 block from the image.
block=block-128;
%Shifting pixel values around zero to improve DCT efficiency.
dctBlock=dct2(block);
quantBlock=round(dctBlock./Q);
%Quantizing the DCT coefficients which causes lossy compression.
dequantBlock=quantBlock.*Q;
%Reversing quantization step for decompression during reconstruction.
idctBlock=idct2(dequantBlock);
idctBlock=idctBlock+128;
%Shifting pixel values back to the original intensity range.
reconstructed(i:i+7,j:j+7)=idctBlock;
%Placing the reconstructed block back into the output image.
end
end
reconstructed=uint8(reconstructed);
%Converting reconstructed image back into uint8 format for display.
figure;
imshow(uint8(I));
title("Original Image");
%Displaying the input image before compression.
figure;
imshow(reconstructed);
```

```
title("Reconstructed Image After High JPEG Compression");
%Displaying the decompressed output image after DCT compression
```

Published with MATLAB® R2025b