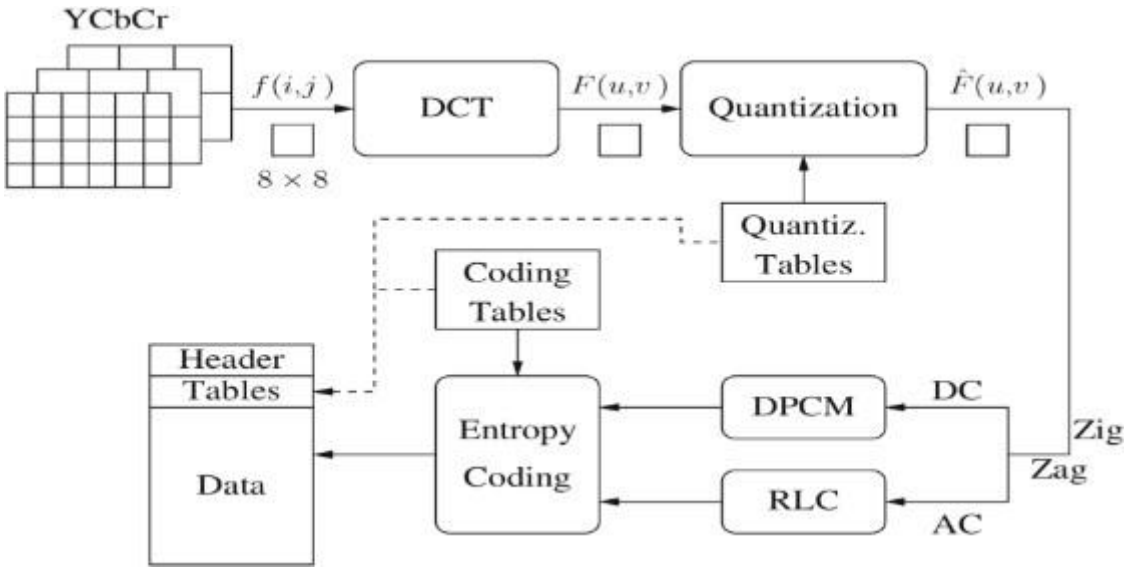
EXPERIMENT NO: 6

AIM: implementation of the jpeg compression on the coloured image.

Software Used: MATLAB R2024b.

Theory:

JPEG (Joint Photographic Experts Group): compression is a widely used lossy compression technique for images. It exploits the limitations of human vision, particularly in perceiving high-frequency color information, to reduce file size while maintaining acceptable visual quality.



colour Space Conversion and Down_sampling: JPEG typically works in the YCbCr color space rather than RGB because the human eye is more sensitive to brightness (luminance, Y) than color (chrominance, Cb and Cr). The chrominance channels (Cb and Cr) can be subsampled without significantly affecting perceived image quality.

- **4:2:0 Subsampling**: In this approach, the chrominance components (Cb and Cr) are downsampled by averaging values from a 2×2 block of pixels. This reduces the chrominance data by a factor of 4 while preserving full resolution in the luminance channel (Y).

Block-wise Processing (8×8 Macroblocks)

- The image is divided into non-overlapping 8×8 blocks for all three channels (Y, Cb, and Cr).
- Each block is processed independently to take advantage of spatial frequency transformations.

Level Shifting: Since the Discrete Cosine Transform (DCT) operates best on values centered around zero, all pixel values are level-shifted by subtracting 128

Discrete Cosine Transform (DCT): The 2D DCT is applied to each 8×88 \times 88×8 block, transforming spatial domain data into frequency domain data.

$$G(u,v) = \frac{1}{4}C(u)C(v)\sum_{x=0}^{7}\sum_{y=0}^{7} g(x,y)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

where:

$$C(u) = \begin{cases} \frac{1}{\sqrt{2}}, & u = 0 \\ 1, & u > 0 \end{cases}$$

Quantization: Quantization reduces precision, allowing for efficient storage. Quantization is the primary source of data loss in JPEG compression. The quantization matrices control the trade-off between compression and image quality **quality factor (QF)** is used to adjust compression levels.

$$\text{quantLuma} = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}$$

$$\text{quantChroma} = \begin{bmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{bmatrix}$$

$$SF = \begin{cases} \frac{100-QF}{50}, & QF \geq 50 \\ \frac{50}{QF}, & QF < 50 \end{cases}$$

The quantization matrix is scaled as

$$Q_s = SF \times Q$$

The quantized coefficients are obtained as

$$B(x,y) = \text{round}\left(\frac{G(x,y)}{Q_s(x,y)}\right)$$

Dequantization: The quantized values are multiplied back by the quantization matrix:

$$G'(u,v) = B(x,y) \times Q_s(x,y)$$

Inverse Discrete Cosine Transform (IDCT): The IDCT reconstructs the spatial domain block

$$g'(x,y) = \frac{1}{4}\sum_{u=0}^{7}\sum_{v=0}^{7} C(u)C(v)G'(u,v)\cos\left(\frac{(2x+1)u\pi}{16}\right)\cos\left(\frac{(2y+1)v\pi}{16}\right)$$

Image Reconstruction

Chrominance Upsampling: The subsampled Cb and Cr components are upsampled to match the size of the Y component using bilinear interpolation (imresize).
YCbCr to RGB Conversion: The processed Y, Cb, and Cr components are combined and converted back to the RGB color space using ycbcr2rgb.
Error Metrics: MSE and PSNR

- **Mean Squared Error (MSE)**: Measures the average squared difference between the original and reconstructed image
- **Peak Signal-to-Noise Ratio (PSNR)**: Evaluates the quality of the compressed image **Compression Ratio:** Defined as the ratio of the original image size to the compressed image size.

$$MSE = \frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[I(i,j) - I'(i,j)]^2$$

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right)$$

Result:

original Image

down$_s$ampled

QF: 10 ,MSE: 84.9543
PSNR: 18.2209 dB
Compression Ratio: 32.2132

QF: 30 ,MSE: 72.8828
PSNR: 20.8582 dB
Compression Ratio: 10.5779

QF: 60 ,MSE: 57.6148
PSNR: 23.8951 dB
Compression Ratio: 5.5596

QF: 90 ,MSE: 15.6789
PSNR: 29.3058 dB
Compression Ratio: 5.2862

original Image

down$_s$ampled

QF: 10 ,MSE: 31.8253
PSNR: 27.2913 dB
Compression Ratio: 75.3384

QF: 30 ,MSE: 17.1406
PSNR: 31.0025 dB
Compression Ratio: 40.5071

QF: 60 ,MSE: 12.1371
PSNR: 32.8564 dB
Compression Ratio: 26.4839

QF: 90 ,MSE: 7.2188
PSNR: 35.7908 dB
Compression Ratio: 12.1546

Discussion:

- **Effect of Quality Factor (QF) on Image Quality & Compression:**
  - **QF = 90 (High Quality, Low Compression):** Minimal artifacts, sharp details, High PSNR low MSE, Large file size, lower compression ratio.
  - **QF = 60 (Moderate Compression, Noticeable Artifacts):** Edges slightly blurred, mild blockiness, higher compression ratio.
  - **QF = 30 (High Compression, Visible Artifacts):** Blocky appearance, noticeable loss of colour details.
  - **QF = 10 (Very High Compression, Poor Quality):** Heavy block artifacts, severe detail loss.
- **Chroma Subsampling (4:2:0):**
  - Reduces color information to save space while preserving perceived image quality.
  - Slight blurring in color details, but minimal impact on natural images.

- **DCT and Quantization:**
  - **DCT transforms** image into frequency components.
  - **Quantization removes** high-frequency details based on QF.
  - **Higher QF** → Smaller quantization values, preserving more details.
  - **Lower QF** → Larger quantization values, discarding high-frequency details.
  - **Y Channel Quantization** affects brightness details more, while **Cb & Cr Quantization** impacts colour fidelity.
- **MSE, PSNR, and Compression Ratio:**
  - **Higher QF → Lower MSE, Higher PSNR.**
  - **Lower QF → Higher MSE, Lower PSNR.**
  - Compression ratio increases at lower QF due to more data being discarded.

- **Best QF Choice:**
  - **QF 85-90:** Best for high-quality images.
  - **QF 50-75:** Good balance between quality and size for web usage.
  - **QF 10-35:** Aggressive compression with noticeable artifacts.

Conclusion:

The quantization matrix plays a central role in JPEG compression by controlling the trade-off between compression ratio and image quality. By scaling the quantization matrix based on the Quality Factor (QF), the experiment demonstrates how different levels of quantization affect the visual quality and compression efficiency of the image. Understanding the effect of the quantization matrix is essential for optimizing JPEG compression and achieving the desired balance between file size and image quality.

Code:

```
clc; clear; close all;
image = imread("peacock.jpg");
I2 = rgb2ycbcr(image);
Y = double(I2(:,:,1));
Cb = double(I2(:,:,2));
Cr = double(I2(:,:,3));
subCb = imresize(Cb, 0.5, 'bilinear');
subCr = imresize(Cr, 0.5, 'bilinear');
cb_u = imresize(subCb, 2, 'bilinear');
cb_r= imresize(subCr, 2, 'bilinear');
down_image = ycbcr2rgb(uint8(cat(3,Y,cb_u,cb_r)));
figure();
subplot(2,3,1); imshow(image); title("original Image");
```

```
subplot(2,3,2); imshow(down_image); title("down_sampled");
Y = Y -128 ; subCr = subCr -128 ; subCb = subCb -128;
Qy= [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];

Qc = [17 18 24 47 99 99 99 99;
         18 21 26 66 99 99 99 99;
         24 26 56 99 99 99 99 99;
         47 66 99 99 99 99 99 99;
         99 99 99 99 99 99 99 99;
         99 99 99 99 99 99 99 99;
         99 99 99 99 99 99 99 99;
         99 99 99 99 99 99 99 99];
[m, n] = size(Y);
[mc, nc] = size(subCb);
block_size = 8;
rows = block_size * floor(m / block_size);
cols = block_size * floor(n / block_size);
rowsc = block_size * floor(mc / block_size);
colsc = block_size * floor(nc / block_size);
qf_values = [10, 30, 60 , 90 ];

for idx = 1:4
   QF = qf_values(idx);
   sf = (QF >= 50) * ((100 - QF) / 50) + (QF < 50) * (50 / QF);
   qsy =max(1,round(sf * Qy)) ;
   qscbcr =max(1,round(sf * Qc)) ;
   Y = Y(1:rows, 1:cols);
   subCb = subCb(1:rowsc, 1:colsc);
   subCr = subCr(1:rowsc, 1:colsc);
   blocks_Y = mat2cell(Y, repmat(block_size, 1, rows / block_size), repmat(block_size, 1, cols / block_size));
   blocks_Cb = mat2cell(subCb, repmat(block_size, 1, rowsc / block_size), repmat(block_size, 1, colsc /
block_size));
   blocks_Cr = mat2cell(subCr, repmat(block_size, 1, rowsc / block_size), repmat(block_size, 1, colsc /
block_size));

   for i = 1:size(blocks_Y, 1)
      for j = 1:size(blocks_Y, 2)
         block_Y = blocks_Y{i, j};
         gY{i, j} = dct2(block_Y);
         bY{i, j} = round(gY{i, j} ./ qsy);
         gcapY{i, j} = bY{i, j} .* qsy;
         g_capY = gcapY{i, j};
         fcapY{i, j} = idct2(g_capY) + 128;
      end
   end

   for i = 1:size(blocks_Cb, 1)
      for j = 1:size(blocks_Cb, 2)
         block_Cb = blocks_Cb{i, j};
         gCb{i, j} = dct2(block_Cb);
         bCb{i, j} = round(gCb{i, j} ./ qscbcr);
         gcapCb{i, j} = bCb{i, j} .* qscbcr;
         g_capCb = gcapCb{i, j};
         fcapCb{i, j} = idct2(g_capCb) + 128;
      end
```

```
    end

    for i = 1:size(blocks_Cr, 1)
        for j = 1:size(blocks_Cr, 2)
            block_Cr = blocks_Cr{i, j};
            gCr{i, j} = dct2(block_Cr);
            bCr{i, j} = round(gCr{i, j} ./ qscbcr);
            gcapCr{i, j} = bCr{i, j} .* qscbcr;
            g_capCr = gcapCr{i, j};
            fcapCr{i, j} = idct2(g_capCr) + 128;
        end
    end

    Y1 = cell2mat(fcapY);
    Cb1 = cell2mat(fcapCb);
    Cr1 = cell2mat(fcapCr);

    Cbre = imresize(Cb1, [size(Y1, 1), size(Y1, 2)], 'bilinear');
    Crre = imresize(Cr1, [size(Y1, 1), size(Y1, 2)], 'bilinear');

    I3 = cat(3, Y1, Cbre, Crre);
    I3 = ycbcr2rgb(uint8(I3));
    original_size = numel(image);
    bY_mat = cell2mat(bY);
    bCb_mat = cell2mat(bCb);
    bCr_mat = cell2mat(bCr);
    quantized_size = nnz(bY_mat) + nnz(bCb_mat) + nnz(bCr_mat);
    compression_rate = original_size / quantized_size;
    mse = mean((image(:) - I3(:)).^2);
    psnr_value = psnr(I3, image);
    subplot(2,3,idx+2), imshow(I3);
    title(sprintf('QF: %d ,MSE: %.4f\nPSNR: %.4f dB\nCompression Ratio: %.4f',QF, mse, psnr_value,
compression_rate));
end
```