#HUFFMAN_ENCODING:

```
txt = 'aaabbcd'; sym = unique(txt);
freq = histc(txt,sym);
prob = freq/sum(freq); sym_num = double(sym);
dict = huffmandict(sym_num,prob);
disp(dict) input = 'aaabbcd';
encod = huffmanenco(double(input),dict);
disp(encod)
deco = huffmandeco(encod,dict);
deco = char(deco);
disp(deco)
disp(length(input)*8)
disp(length(encod))
disp(length(encod)/(length(input)*8));
```

#LEMPEL ZIV ENCODING:

```
function encoded = lempel_ziv_encode(input)

    input = char('is this is');
    n = length(input);
    encoded = {};
    dictionary = {};
    i = 1;
    while i <= n
        prefix = '';
        j = i;
        while j <= n && any(strcmp(dictionary, [prefix, input(j)]))
            prefix = [prefix, input(j)];
            j = j + 1;
        end
        if isempty(prefix)
            encoded{end + 1} = [0, input(i)];
        else
            prefixIndex = find(strcmp(dictionary, prefix));
            if j <= n
                encoded{end + 1} = [prefixIndex, input(j)];
            else
                encoded{end + 1} = [prefixIndex, ''];
            end
        end
        if j <= n
            dictionary{end + 1} = [prefix, input(j)];
        end
        i = j + 1;
    end
end
```

#FOURIER

```
X = [1 2 ;
     3 4];
[n, ~] = size(X);
disp(n)
F = zeros(n,n);
%D = zeros(n,n);
for u = 0:n-1
   for v = 0:n-1
      sum = 0;
      for x = 0:n-1
         for y = 0:n-1
            expTerm = exp(-1j *2*pi*((u*x)+(v*y))/n);
            sum = sum + X(x+1,y+1)*expTerm;
         end
      end
      F(u+1,v+1) = sum;
   end
end
disp(real(F))
```

#DCT

```
X = [1 2;
     3 4];
[n,~]= size(X);
D = zeros(n,n);
alpha = @(k) sqrt(1/n)*(k==0) + sqrt(2/n)*(k>0);
for u = 0:n-1
   for v = 0:n-1
      sum = 0;
      for x = 0:n-1
         for y = 0:n-1
            cos1 = cos(((2*x + 1)*u*pi)/(2*n));
            cos2 = cos(((2*y + 1)*v*pi)/(2*n));
            sum = sum + X(x+1,y+1)*cos1*cos2;
         end
      end
      D(u+1,v+1)= alpha(u)*alpha(v)*sum;
   end
end
disp(D)
```

```
#JPEG
img = imread('football.jpg');
grayImg = rgb2gray(img);
grayImg = double(grayImg) - 128;
[m, n] = size(grayImg);
paddedImg = padarray(grayImg, [mod(8 - mod(m, 8), 8), mod(8 - mod(n, 8), 8)], 'replicate',
'post');
[paddedM, paddedN] = size(paddedImg);
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];
compressedImg = zeros(paddedM, paddedN);
for i = 1:8:paddedM
   for j = 1:8:paddedN

       block = paddedImg(i:i+7, j:j+7);

       dctBlock = dct2(block);

       quantizedBlock = round(dctBlock ./ Q);

       dequantizedBlock = quantizedBlock .* Q;

       compressedBlock = idct2(dequantizedBlock);

       compressedImg(i:i+7, j:j+7) = compressedBlock;
   end
end

compressedImg = compressedImg(1:m, 1:n);
compressedImg = compressedImg + 128;
compressedImg = uint8(compressedImg);
%imwrite(compressedImg, 'compressed_image.jpg');
imshow(compressedImg);
title('Compressed Image');
```

#BASIC AUDIO COMPRESSION

```
[audio, fs] = audioread('audio.mp3');
audio = audio(:, 1);
compression_factor = 5; downsampled_audio = audio(1:compression_factor:end);
num_bits = 8;
quantized_audio = round(downsampled_audio * (2^(num_bits - 1))) / (2^(num_bits - 1));
audiowrite('compressed_audio.wav', quantized_audio, fs / compression_factor);
disp('Playing original audio...');
sound(audio, fs);
pause(length(audio) / fs + 2);
disp('Playing compressed audio...');
sound(quantized_audio, fs / compression_factor);
```

#BASIC VIDEO COMPRESSION

```
videoReader = VideoReader('video.mp4');
outputFile = 'compressed_video.mp4';
videoWriter = VideoWriter(outputFile, 'MPEG-4');
open(videoWriter);
frame_skip = 4; color_bits = 2;
frame_idx = 1;
while hasFrame(videoReader)
    frame = readFrame(videoReader);
    if mod(frame_idx, frame_skip) == 0
        quantized_frame = floor(double(frame) / (2^(8 - color_bits))) * (2^(8 - color_bits));
        quantized_frame = uint8(quantized_frame);
        writeVideo(videoWriter, quantized_frame);
    end
    frame_idx = frame_idx + 1;
end
close(videoWriter);
disp('Video compression complete. Compressed video saved as compressed_video.mp4');
implay('compressed_video.mp4');
```