```
clc
clear
close all
img = imread('lena.tif');
% img_double = im2double(img);
% img_uint8 = uint8(img_double * 255);
% img_vector = reshape(img_uint8, 1, []);
% img_bits = dec2bin(img_vector, 8);
%reshape(img_bits.', 1, []);
img_bitstream =
'111111111111111000000000000000000001111'
input = str2num(img_bitstream')'
input = 1 \times 38
                                 1
                                                               1 ...
   1 1
max_rep = find_max_consecutive_repetition(input) %max_repetition = 23
max_rep = 19
bit_len = ceil(log2(max_rep)) %bit_len = 5
bit_len = 5
rle = rle_encode_bitstream(input,bit_len)
rle =
'011111100110001001'
compression_ratio = size(input)/size(rle)
compression_ratio = 2.1077
function max_rep = find_max_consecutive_repetition(bitstream)
   maxCount = 0;
    currentCount = 0;
   previousBit = NaN; % initialize with NaN to handle the case where the
bitstream starts with a NaN value
    for i = 1:length(bitstream)
       bit = bitstream(i);
        if bit == previousBit
            currentCount = currentCount + 1;
        else
           maxCount = max([maxCount, currentCount]);
            currentCount = 1;
           previousBit = bit;
        end
    end
```

```
max_rep = max([maxCount, currentCount]);
end
function rle = rle_encode_bitstream(bitstream,bit_len)
    % Initialize variables
   rle = [];
    current = bitstream(1);
    count = 1;
   % Loop through the bitstream
   for i = 2:length(bitstream)
        if bitstream(i) == current
            % If the current bit is the same as the previous one, increment
count
            count = count + 1;
        else
            % If the current bit is different, add the previous bit and its
count to the output
            rle = [rle dec2bin(count,bit_len) num2str(current)];
            current = bitstream(i);
            count = 1;
        end
    end
    % Add the last bit and its count to the output
   rle = [rle dec2bin(count,bit_len)];
   rle = [rle num2str(current)];
end
```