PART-1

This is a programming assignment which requires you to encode and decode binary message bits using repetition codes and arithmetic codes.

Input: A text file of size 1 KB of your choice.

Computing Environment: Matlab / Python without using built-in libraries for arithmetic code/repetition code

Experiment 1:

1) Read the input text-file and convert it to a binary string, say of length M bits.

2) Generate a random binary error pattern of length M with hamming weight d such that the non-zero entries are uniformly distributed across M bits.

3) XOR the above error pattern with the message bits to obtain a new sequence denoted by y.

4) Using y, retrieve the text-file without any error detection/ correction coding.

5) In the decoded file, compute the percentage of modified characters with respect to the input file.

6) Repeat the above experiment by varying the value of d = {10, 100, 200, 500, 5000}.

Experiment 2:

1) Read the input text-file and convert it to a binary string, say of length M bits.

2) Divide the input string into several chunks such that each chunk is of size k bits.

3) Encode each chunk into a sequence of n bits by using (i) Huffman, (ii) (4-bit per symbol) Extended Huffman, and repetition codes and (iii) Arithmetic codes. After encoding, let the total number of bits generated from the entire text-file be M′.

4) Generate a random binary error pattern of length M′ with Hamming weight d such that the non-zero entries are uniformly distributed across M′ bits.

5) XOR the above error pattern with the encoded bits to obtain a new sequence denoted by y.

6) Using y, retrieve the text-file by decoding.

7) Compute the number of errors you could detect, and the number of errors you could correct.

8) In the decoded file, compute the percentage of modified characters with respect to the input file.

9) Repeat the above experiment by varying the value of d = {10, 100, 200, 500, 5000}.

10) Compare the results between the three coding techniques and experiment-1 without any such technique.

PART-2

This is a programming assignment which requires you to apply Discrete Wavelet Transform on an image and see how much compression you can achieve without a prominent loss.

Input:

1) A gray-scale png image (512x512 matrix) of your choice.

2) Levels of decomposition

3) Threshold/s for different sub-bands

Computing Environment: Matlab / Python

Experiment:

1) Read the input image-file and convert it to a matrix.

2) Take the 2-D DWT of the image.

3) Having found the DWT coefficients of the image, compute the average energy of each sub-band. Now, we are going to "prune" some of the coefficients in each sub-band. Keep those with the highest energy, by selecting as sub-band specific threshold and discard the rest. Experiment with different values for the threshold and see how the reconstructed image looks after taking the 2-D inverse DWT.

4) Next, vary the number of coefficients retained and see how many coefficients you can discard until the image degradation is perceptually significant.

NOTE: Use of any already existing calibration toolbox and library is prohibited.Students shall use libraries for computing mathematical functions and evaluations, and performing operations (such as read/write/ matching, etc) only.Students can form groups of upto 2 students. Please fill the sheet shared as soon as possible if not already done.

Submissions will be done on moodle in the form of a pdf report and a zip containing source code(with compiled source).
The deadline for this assignment is 11:59 pm @ 4 February 2022.