Problem 1

1. -
2. Observations for an 8 X 8 sub window whose top left corner is (420, 45):
   a. When we apply DCT to transform the image into frequency(cosine) domain we find the matrix as follows:

```
1.0e+03 *

 1.1061    0.4761   -0.1196   -0.0859    0.0509    0.0159   -0.0245   -0.0014
-0.0006    0.0113   -0.0089    0.0156   -0.0030   -0.0083    0.0038   -0.0066
-0.0118   -0.0055    0.0176   -0.0058   -0.0167    0.0176   -0.0001   -0.0066
-0.0077    0.0110    0.0009   -0.0084   -0.0000    0.0023   -0.0033   -0.0026
 0.0044   -0.0052    0.0019    0.0006    0.0056   -0.0025    0.0018   -0.0003
-0.0034    0.0012   -0.0040    0.0028   -0.0006   -0.0030   -0.0031   -0.0037
 0.0026    0.0078   -0.0061    0.0047   -0.0059    0.0028   -0.0059    0.0013
 0.0002    0.0028    0.0011   -0.0005    0.0043    0.0026    0.0077    0.0007
```

   We can clearly see that the values have a high order of magnitude.

   b. Quantizing the above matrix results in the following matrix:

```
35    22    -6    -3     1     0     0     0
 0     0     0     0     0     0     0     0
 0     0     1     0     0     0     0     0
 0     0     0     0     0     0     0     0
 0     0     0     0     0     0     0     0
 0     0     0     0     0     0     0     0
 0     0     0     0     0     0     0     0
 0     0     0     0     0     0     0     0
```

   The values become much smaller in magnitude and there is a lot of redundancy which can be taken advantage of in any standard compression algorithm (like Huffman Encoding or Run Length Encoding).

   c. Dequantizing the above Quantized matrix gives us the following:

```
1120    484   -120    -96     48      0      0      0
   0      0      0      0      0      0      0      0
   0      0     32      0      0      0      0      0
   0      0      0      0      0      0      0      0
   0      0      0      0      0      0      0      0
   0      0      0      0      0      0      0      0
   0      0      0      0      0      0      0      0
   0      0      0      0      0      0      0      0
```

   We do not exactly get back the matrix in (a) as expected. We have lost a lot of information. But the fact is that we do not require values in this domain to be precise as long as the image can be reconstructed precisely.

d. Reconstructing the image, we get the following matrix for the 8 X 8 sub window:

203.0354	203.1618	203.4685	184.8903	132.6498	75.1105	54.2592	63.4246
199.0354	201.5049	205.1254	188.8903	136.6498	76.7673	52.6024	59.4246
193.3785	199.1618	207.4685	194.5471	142.3066	79.1105	50.2592	53.7677
189.3785	197.5049	209.1254	198.5471	146.3066	80.7673	48.6024	49.7677
189.3785	197.5049	209.1254	198.5471	146.3066	80.7673	48.6024	49.7677
193.3785	199.1618	207.4685	194.5471	142.3066	79.1105	50.2592	53.7677
199.0354	201.5049	205.1254	188.8903	136.6498	76.7673	52.6024	59.4246
203.0354	203.1618	203.4685	184.8903	132.6498	75.1105	54.2592	63.4246

whereas the original image matrix was:

196	202	195	192	126	76	55	42
195	198	197	189	133	78	57	51
199	200	199	194	149	83	54	43
193	201	193	195	157	80	52	55
200	195	202	197	149	71	49	60
192	204	201	197	139	65	59	45
191	193	198	192	127	67	60	58
193	192	199	190	127	72	74	62

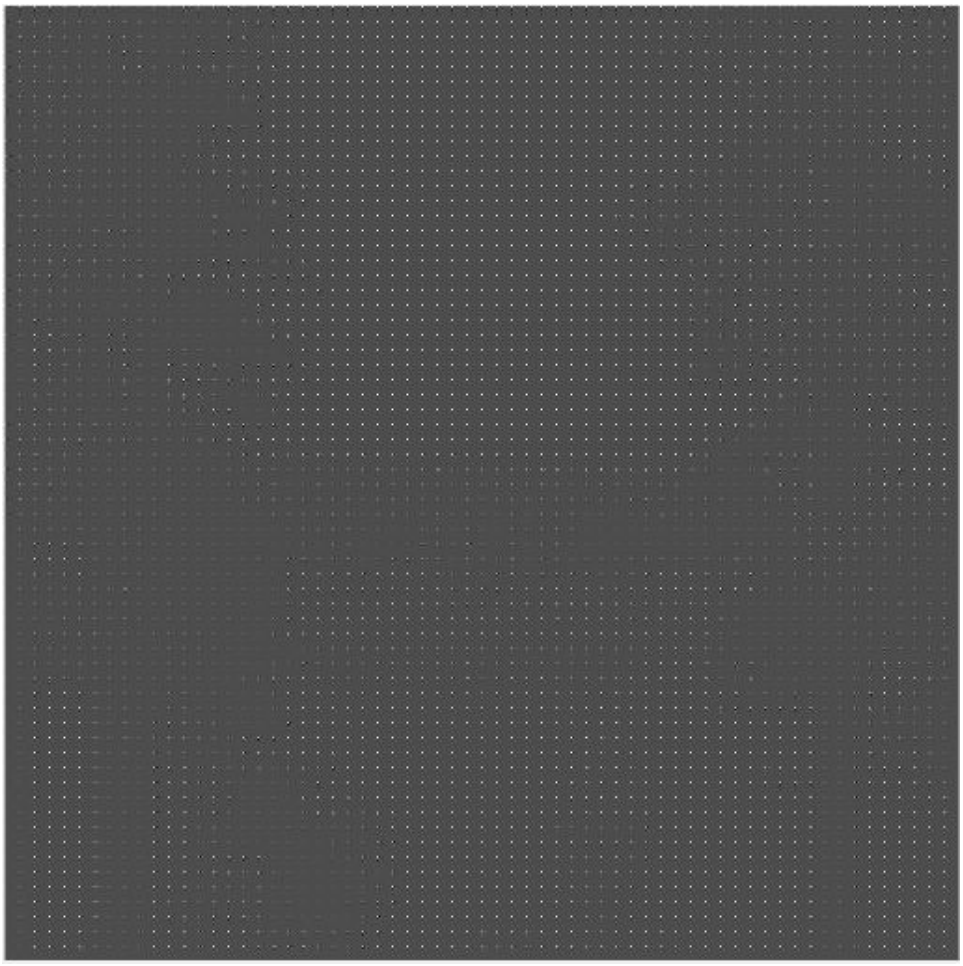
Here is the fantastic thing. Though we had so many zeros in (c) we actually got back pretty accurate values for the image in those positions. As a result, transforming to frequency domain using cosine transform helped us to take advantage of redundant values at the same time allowing us to reconstruct the image so precisely. This is the beauty of what change of perspective can bring!

Finally, the images:

Original Image

Reconstructed Image

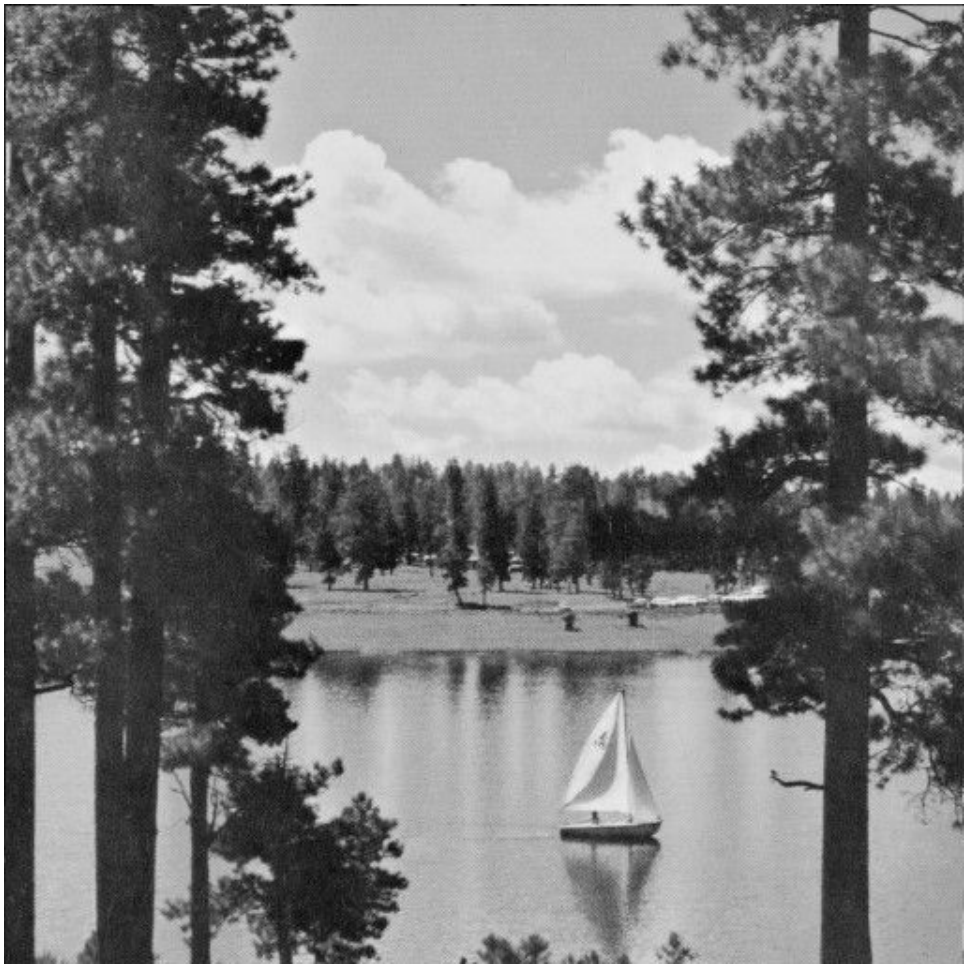
3.



This is how the image Lake.TIF looks like when we apply DCT (and quantize) each of the 8 X 8 sub windows of the image.

One of the most important observations of the matrix values corresponding to the above image is that it is a very sparse matrix with a lot of zeros (redundancy) and as a result most of the standard compression algorithms like Huffman Encoding, Run Length Encoding and Arithmetic compression will do a really good job in compressing the redundancies.

4.



Original Image

Reconstructed image with compression factor (c) = 2 having RMSE = 7.4132 and Entropy = 0.0010167

Reconstructed image with compression factor(c) = 5 having RMSE = 10.3763 and Entropy = 0.0034871

It is for this value of c = 5 that the distortions of the reconstructed image are just perceptible.

Reconstructed image with compression factor(c) = 10 having RMSE = 14.0217 and Entropy = 0.010267

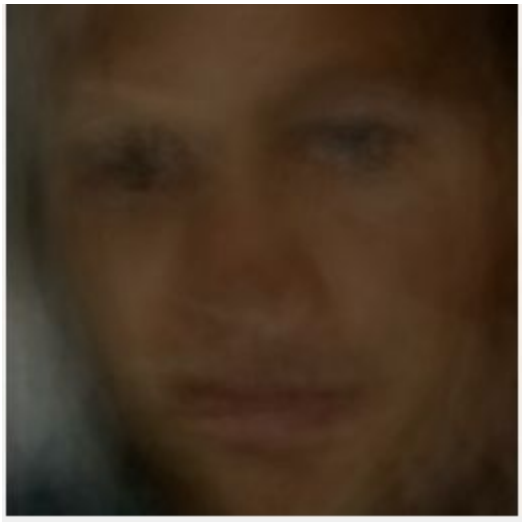
Explanation: When we use a large c, while doing the quantization step we divide by a larger value and hence lose more information in the process which affects our reconstruction. As a result the RMSE error goes up and interestingly entropy also increases. The increase of entropy can be explained by the fact that the pixel values converge more and more. For example, initially if pixel values 130 and 140 could be reconstructed differently now both will take a single value. This is evident from the fact that the white shadows on the lake are aliased and so are the clouds. This aliasing is very prominent. Since, many pixels share the same value now, the net effect of the formula $\Sigma - P_i * \log(P_i)$ is an increase due to the multiplication operation.

Problem 2

1.

Original Image(000_017)

Image(000_017) reconstructed with 256 Principal Components



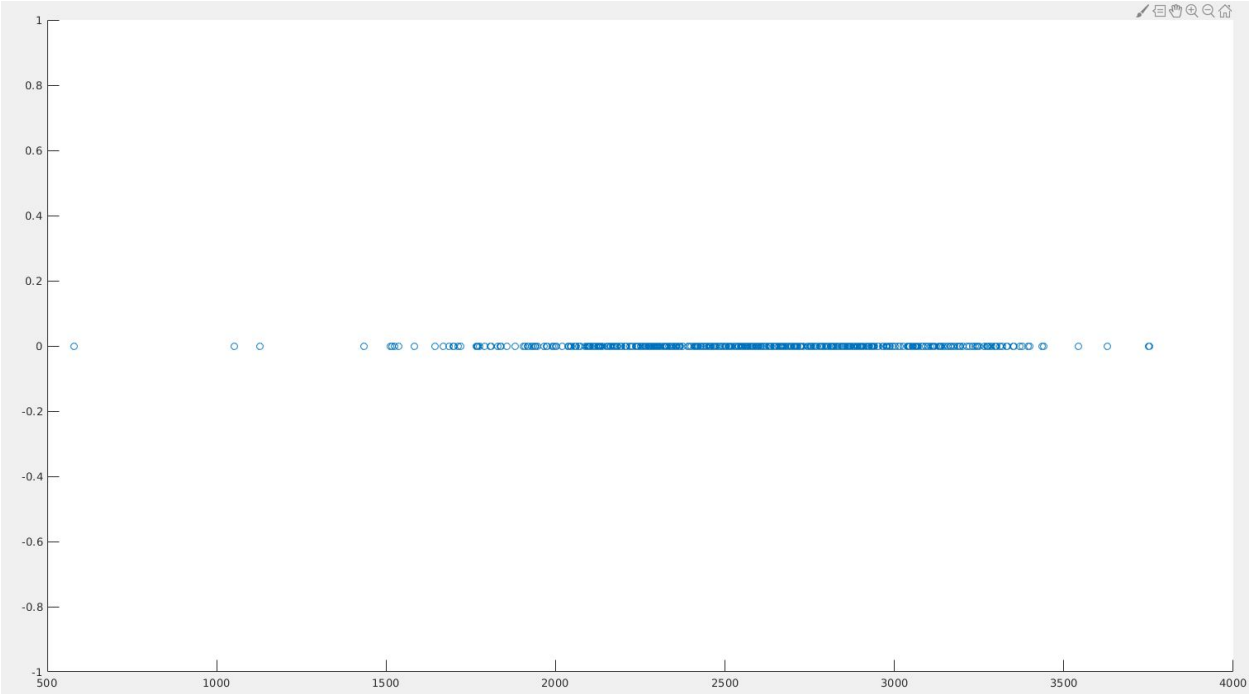
Image(000_017) reconstructed with 35 Principal Components

Original Image(000_001)

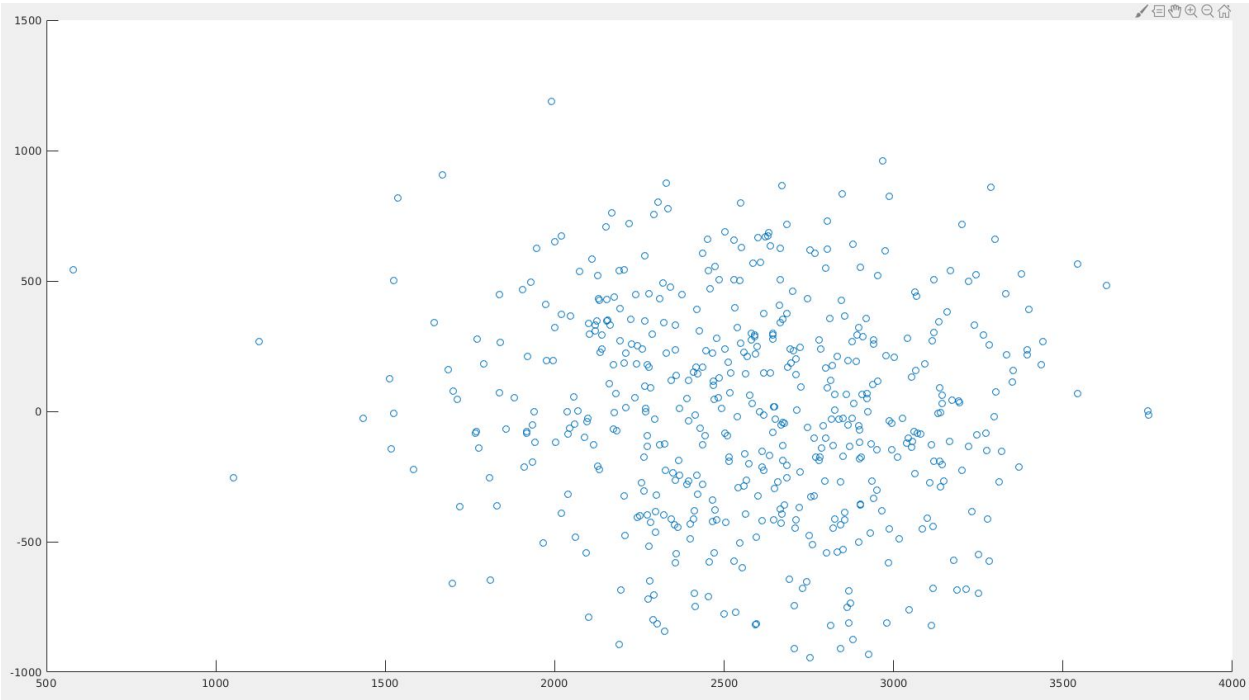
Image(000_001) reconstructed with 256 Principal Components

Image(000_001) reconstructed with 35 Principal Components

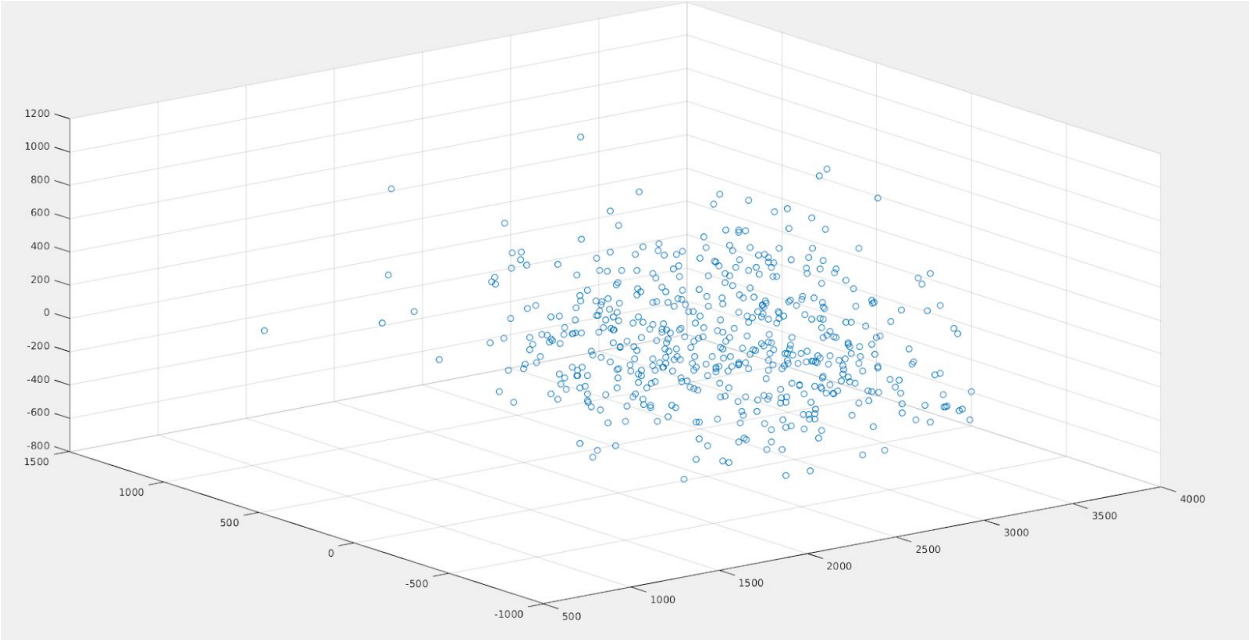
2.



This is how the images are clustered in 1D



This is how the images are clustered in 2D



This is how the images are clustered in 3D

Algorithm:

Step1:

Find the covariance matrix. Covariance matrix C is a matrix where the entry at the ith row and jth column contains the covariance between the ith and jth dimension.

Step2:

We find the Eigenvectors and Eigenvalues for the matrix C.

Step3:

We take the 35 Eigenvectors corresponding to the highest 35 Eigenvalues and create a matrix D, where the columns of the matrix are the eigenvectors.

Step4:

Computing R = X * D (X = Matrix whose rows contain the images flattened) summarises the images using the required number of principal components. The resulting matrix R has rows containing the images summarised with required number of Principal Components.

Step5:

To reconstruct the images we compute *images* = R * D'. Now images is a matrix whose rows

contain the images flattened with the original number of dimensions (but obviously we have lost information).

The Covariance matrix can be computed using the formula:

$$\frac{1}{N} X^\top X$$

This will result in a Covariance matrix C of dimensions 196608 X 196608 as each image has 196608 dimensions. Due to memory constraints it is not feasible to compute this large matrix. Therefore, we compute the Gram Matrix, given by the formula:

$$\frac{1}{N} X X^\top$$

which results in a matrix of dimensions 520 X 520 (since number of images = 520). We compute the Eigenvectors of this matrix and then we use mathematical tricks to convert the Eigenvectors of the Gram matrix into corresponding Eigenvectors of the Covariance matrix.

Taken from stats.stackexchange.com website:

The section is about the relationship between the eigenvectors of covariance matrix $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$ and the eigenvectors of the Gram matrix $\frac{1}{N}\mathbf{X}\mathbf{X}^\top$ (in the context of PCA). Let $\mathbf{v}_i$ be a unit-length eigenvector of $\frac{1}{N}\mathbf{X}\mathbf{X}^\top$:

$$\frac{1}{N}\mathbf{X}\mathbf{X}^\top\mathbf{v}_i = \lambda_i\mathbf{v}_i.$$

If we multiply this equation by $\mathbf{X}^\top$ from the left:

$$\frac{1}{N}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{v}_i) = \lambda_i(\mathbf{X}^\top\mathbf{v}_i),$$

we see that $\mathbf{X}^\top\mathbf{v}_i$ is an eigenvector of $\frac{1}{N}\mathbf{X}^\top\mathbf{X}$.

However, it will not have unit length! Indeed, let us compute its length:

$$\|\mathbf{X}^\top\mathbf{v}_i\|^2 = (\mathbf{X}^\top\mathbf{v}_i)^\top\mathbf{X}^\top\mathbf{v}_i = \mathbf{v}_i^\top\mathbf{X}\mathbf{X}^\top\mathbf{v}_i = \mathbf{v}_i(N\lambda v_i) = N\lambda\|\mathbf{v}_i\|^2 = N\lambda_i.$$

So the squared length of $\mathbf{X}^\top\mathbf{v}_i$ is equal to $N\lambda_i$. Therefore, if we want to transform $\mathbf{v}_i$ into a unit-length covariance matrix eigenvector $\mathbf{u}_i$, we need to normalize it have unit length:

$$\mathbf{u}_i = \frac{1}{(N\lambda_i)^{1/2}}\mathbf{X}^\top\mathbf{v}_i.$$