This homework assignment has three parts, two are required and one is bonus.

**PART 1:**

Implement the k-means algorithm to cluster the image pixels. Your algorithm should take the pixels of an RGB image, group the pixels into k clusters, and output the clustering vectors as well as the labels of the clustered pixels. In the RGB space, each pixel is represented with 3 colors, which are in between 0 and 255.

To test your algorithm, use the image (sample.jpg) provided on the course website (of course, you can use other images to obtain additional results). Run your implementation at least for the following values of k = {2, 4, 8, 16}. For each of these k values,

- Report the clustering error
- Give the values of the clustering vectors
- Give the clustered image. To obtain this clustered image, represent each pixel with the RGB values of its clustering vector.

Additionally, using your implementation, determine the optimum k value for this image. For that, do NOT just consider the k values given above (i.e., do NOT just consider k = {2, 4, 8, 16}). Explain how you determine the optimum value. Moreover, for your optimum k value, report the clustering error and give the clustered image, as explained above.

In your report, give all details of your implementation, such as how to select the initial values and the stopping criterion. Your report should also include the aforementioned outputs. Since the outputs include the clustered images, submit a color printout of your report.

In this assignment, you may use any programming language you would like. If you do not know how to process images in your preferred programming language, you can employ the following Matlab codes. Of course, if you do not want to use the following codes, that is completely ok as long as you process the image pixels and give the required outputs.

```matlab
% Reads sample.jpg into an im matrix, whose dimensions are N, M, and 3
im = imread('sample.jpg');

% Reshapes the 3D im matrix into a 2D matrix, called im2, whose
% dimensions are NxM and 3. In this 2D matrix, each row corresponds to
% a pixel, and the columns correspond to red, green, and blue channels
% in the RGB color space, respectively
im2 = reshape(im, [size(im,1) * size(im,2) 3]);

% You need to run the k-means algorithm on the im2 matrix. Your
% k-means algorithm should output the clustering vectors (let’s call
% them V) and the labels of the clustered pixels (let’s call them
% cmap). To implement the k-means algorithm, you can use Matlab.
% However, if you find Matlab too slow (and if you do not know how to
% read images in your preferred programming language), you can write
% the contents of im2 into a file and read this file in your program.
% Similarly, you can calculate clustering vectors V and labels cmap
```
in your program, write them into files, and read them from Matlab
to obtain the clustered image

Suppose that your program outputs matrix V, whose dimensions are k
and 3, and vector cmap, whose dimension is NxM. Also suppose that
your labels are in between 1 and k. Then you may use the following
Matlab codes to produce the clustered image
cmap2 = reshape(cmap,[size(im,1) size(im,2)]);
M = V / 255;
classifiedImage = label2rgb(cmap2, M);

Shows the clustered image in Matlab and writes it into a bitmap file
figure, imshow(clusteredImage)
imwrite(clusteredImage,'output.bmp')

PART 2:
Implement an agglomerative hierarchical clustering algorithm to cluster the image pixels. Similarly, your
algorithm should take the pixels of an RGB image, group the pixels into k clusters using this
agglomerative algorithm, and output the clustering vectors as well as the labels of the clustered pixels.

The computational time of this part could be high due to the number of pixels in the image. Thus, propose a technique to overcome this problem. For example, you may “somehow” form small groups of pixels and consider each of these groups as the initial clusters of your agglomerative algorithm. Do not forget that there is not only one correct solution for this part. Thus, try to consider different alternatives.

Test your algorithm on the same image (sample.jpg) and obtain the results for different values of k.
Prepare a report for this second part similar to Part 1, also addressing the same questions given in the first part. Additionally, for this second part, give the details of your technique that you will propose to overcome the computational time problem.

At the end, compare the results that you will obtain in Part 1 and Part 2, and include this comparison into your report.

Similar to the previous assignments, prepare your report neatly and properly. But this time, since it includes the clustered images, submit a color printout of your report. Additionally, email the source code of your implementation. The subject line of your email should be CS 550: HW3. Do not submit the printout of your source code.
**BONUS PART:**

In your second homework, you implemented the backpropagation algorithm, which initialized its weights randomly. Now, implement and use an autoencoder or a restricted Boltzmann machine for pretraining the network layer-by-layer. Then, use the pretrained weights as the initial weights for your backpropagation algorithm. For this part, you have to implement and use your own code. That is, you are NOT allowed to use any code from any deep learning library.

After implementing the pretraining algorithm (either an autoencoder or a restricted Boltzmann machine), apply it to one of the three-hidden layered networks that you used in your second homework. Take your runs only on the MNIST dataset and only for your selected network.

Then prepare a separate report for this part. Your report should include

- The derived update rules for pretraining (either for an autoencoder or a restricted Boltzmann machine) and the loss function that you use in your derivation. It is not necessary to derive these rules by yourselves. Just write down the rules that you use.
- For each pretraining (for each layer), the loss values after each epoch. Do not write the loss values directly but give a plot of these values as a function of the epoch number.
- The training and test set accuracies obtained with the pretrained network.
- The training and test set accuracies obtained with the same network whose initial values are randomly selected (you can use the values that you obtained for your second homework).

For this part, prepare a separate report and submit it separately. Also separately email the source code of your implementation. The subject line should be CS 550: HW3 (bonus). Do not submit the printout of your source code.