Question 4:

1. K-Medoids Implementation:
    - **Initialization:** Randomly choose k datapoints in n samples (n pixels of the image)
    - **Distance:** Euclidian distance
    - **Iteration:**
        - Assign each datapoints to each nearest medoid.
        - Calculate the sum of distance between each datapoint in the same cluster with its cluster medoid. Calculate the sum of distance for all clusters
        - Randomly choose another point within assigned clusters as new medoid, calculate the sum of distance between each data points in the same cluster with new medoid.
        - If the new sum of distance is less than the old sum of distance, assign medoid := new_medoid. Re-iterate from step 1
    - **Stopping criteria:**
        - Max iteration = 500
        - Each assigned cluster remains unchanged after iteration
2. As the number of clusters increases, the compressed image look more like the original image, but also the running time increases.
3. The K-medoid initialization does affect the final result. Choosing K medoids too close to each other causes the image to look bad and it takes longer to run

4. K-Mean Implementation:
    - **Initialization:** Randomly choose k datapoints in n samples (n pixels of the image)
    - **Distance:** Euclidian distance
    - **Iteration:**
        - Assign each datapoints to each nearest center.
        - Calculate the mean of each cluster and assign mean point as new cluster center
    - **Stopping criteria:**
        - Each assigned cluster center remains unchanged after iteration
5. As the number of clusters increases, the compressed image looks more like the original image, but also the running time increases.
6. The K-means initialization does affect the final result. Choosing K mean too close to each other causes the image to look bad and it takes longer to run