Question 1

```
[46] # Use only numpy library
     import numpy as np

     # Randomly initialize centroids at the beginning
     def centroid_init(data, num_samples, k):
         centroids = data[np.random.choice(num_samples, k, replace=False)]
         return centroids

     # Cluster Assignment function
     def distance(data, centroids, num_samples, assignments):
         cluster_dict = dict()
         for i in range(num_samples):
             x_i = 0
             distance = np.linalg.norm(data[i] - centroids[0])
             for j in range(1, k):
                 if distance > np.linalg.norm(data[i] - centroids[j]):
                     x_i = j
                     distance = np.linalg.norm(data[i]-centroids[j])

             assignments[i] = x_i

             if x_i not in cluster_dict:
                 cluster_dict[x_i] = [i]

             else:
                 cluster_dict[x_i].append(i)

         return cluster_dict

     # Move centroid function
     def move_centroid(data, cluster_dict, cluster_c):
         for x, samples in cluster_dict.items():
             cluster_c[x] = np.mean(np.array([data[i] for i in samples]), axis=0)

         return cluster_c
```

```python
def k_means(data, k):
    # Get the number of samples from the breast cancer data
    num_samples = data.shape[0]
    # Initialize centroids randomly
    cluster_c = centroid_init(data, num_samples, k)
    # Create numpy array of zeroes of the number of features of the data
    past_c = np.zeros((k, data.shape[1]))
    # Create a cluster assignment numpy array that's empty of the number of samples of data
    assignments = np.empty(num_samples, dtype=np.int8)

    flag = True
    while flag == True:
        # Initialize a cluster dictionary and assign it to the cluster assignment function
        cluster_dict = dict()
        cluster_dict = distance(data, cluster_c, num_samples, assignments)

        # Assign the cluster centroids to the new moved centroids
        cluster_c = move_centroid(data, cluster_dict, cluster_c)

        # End the while loop if the old centroids are the same as the new centroid assignments
        if (cluster_c == past_c).all():
            flag = False

        # Assign the old centroids to the new centroids after checking if they were equal before
        past_c = np.copy(cluster_c)

    return assignments, cluster_c
```