1, Suppose we have 10 college football teams X1 to X10. We want to cluster them into 2 groups. For each football team, we have two features: One is # wins in Season 2016, and the other is # wins in Season 2017.

Team	# wins in Season 2016 (x-axis)	# wins in Season 2017 (y-axis)
X1	3	5
X2	3	4
X3	2	8
X4	2	3
X5	6	2
X6	6	4
X7	7	3
X8	7	4
X9	8	5
X10	7	6

(1) Initialize with two centroids, (4, 6) and (5, 4). Use Manhattan distance as the distance metric. Please use K-Means to find two clusters.

(2) Initialize with two centroids, (4, 6) and (5, 4). Use Euclidean distance as the distance metric. Please use K-Means to find two clusters.

(3) Initialize with two centroids, (3, 3) and (8, 3). Use Manhattan distance as the distance metric. Please use K-Means to find two clusters.

(4) Initialize with two centroids, (3, 2) and (4, 8). Use Manhattan distance as the distance metric. Please use K-Means to find two clusters.

What are your observations?

2, K-Means Clustering with Real World Dataset

First, download the Iris data set from: https://archive.ics.uci.edu/ml/datasets/Iris

Then, implement the K-means algorithm. I have uploaded a sample python K-means code in Canvas. K-means algorithm computes the distance of a given data point pair. Replace the distance computation function with Euclidean distance, 1- Cosine similarity, and 1 – the **Generalized** Jarcard similarity.

For the generalized Jarcard, you can visit https://en.wikipedia.org/wiki/Jaccard_index for more details. For instance, in the uploaded kmeans.py, there is a function that is called "distance" defined next:

```
def distance(instance1, instance2):
    if instance1 == None or instance2 == None:
    return float("inf")
    sumOfSquares = 0
    for i in range(1, len(instance1)):
    sumOfSquares += (instance1[i] - instance2[i])**2
    return sumOfSquares
```

You can replace this function with different similarity/distance metrics.

Q1: Run K-means clustering with Euclidean, Cosine and Jarcard similarity. Specify K= the number of categorical values of y (the variable of label). Compare the SSEs of Euclidean-K-means Cosine-K-means, Jarcard-K-means. Which method is better and why?

Q2: Compare the accuracies of Euclidean-K-means Cosine-K-means, Jarcard-K-means. First, label each cluster with the label of the highest votes. Later, compute the accuracy of the K-means with respect to the three similarity metrics. Which metric is better and why?

Q3: Which of Euclidean-K-means, Cosine-K-means, Jarcard-K-means requires more iterations and times and why?

Q4: Compare the SSEs of Euclidean-K-means Cosine-K-means, Jarcard-K-means with respect to the following three terminating conditions:

- when there is no change in centroid position
- when the SSE value increases in the next iteration
- when the maximum preset value (100) of iteration is complete

Which method requires more time or more iterations and why?

3, Understanding K-Means:

- List the general idea of K-means clustering algorithm?
- Please give a scenario in which K-means cluster may not work very well?
- What is the advantage of K-Means? What are the disadvantages of K-Means?
- The classic K-means algorithm randomly initializes K centers. Is there any better strategy for selecting K initial centers?

Please submit a report (PDF or word) that includes a link to your code, your answers/results, and your explanations or interpretations (if any).