Programming Assignment 3

k-means Algorithm:

1. Place k points into the feature space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the k centroids.
4. Repeat steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

Pseudo Code:

```
1. make initial guesses for the centres of the clusters
2. until there are no changes in any mean
    a. use the estimated means to classify the samples into clusters
    b. for i from 1 to k
        i. Replace centre with the mean of all the samples for cluster i
    c. end_for
3. end_until
```

where k is number of clusters

Remarks:

The result of k-means depends on initialization – the choice of the initial tentative centroids, even if we know, or have guessed correctly, the number of clusters k. If we start from wrong entities as the tentative centroids, the result can be rather disappointing.

Our Approach and Execution Process:

User enters the value for k, i.e. number of clusters and we select them as initial cluster centres. Then we start implementing the k-means algorithm in which we calculate the distance between each training sample from the identified cluster centres. We update the cluster centres to identify the new cluster centres for each cluster group. We continue to do the same until the cluster groups do not change in the results. Once we get to this point, we stop and assign a class value to the cluster which is the majority class of the training examples in the cluster.

For Execution:

1. Execute the program.
2. Enter the value of k - number of clusters.

Methods used:

1. main() reads the number of clusters and the car data file. It calls the function - newCentroid() to calculate the new cluster centres.
2. newCentroid():
    a. Calculates the new centres of the clusters recursively, the recursion stops if the centres of previous iteration are similar to the current centres.
    b. Assigns class values to each cluster by choosing the majority class of the training examples of that cluster.

    c. Calls the function classificationErrorRate()
3. classificationErrorRate(): Computes the fraction of misclassified samples when compared to the original training example classification.

Confusion Matrix:
Number of clusters = 4

	Predicated Classification				
Actual Classification	N = 1728	unacc	acc	good	vgood
	unacc	1210	0	0	0
	acc	384	0	0	0
	good	69	0	0	0
	vgood	65	0	0	0
		1728	0	0	0

Classification Error Rate: 0.29976851851851855
Accuracy: 0.7002314814814815

Conclusion:
The solution for the car data is not an optimal solution, which can be observed from the output of the program. This occurs mainly because majority of the training samples in the car data are classified as "unacc". Thus, the output will always have a higher error rate.

To minimize the error rate, we can increase the number of cluster creation.