CSE5243 **Student Name:** _____
Autumn 2019 **Student ID:** _____
Programming Assignment 3
Due date: Thursday, Nov 21, 12:44pm

Instructions:

In this programming assignment, you will implement the k-means algorithm. The total point is 108 and you may get some points as bonus. Good luck!

Note: This is not an easy programming assignment. Start earlier!

Grade Table (for TA/grader use only)

Question	Points	Score
1	10	
2	10	
3	88	
4	0	
Total:	108	

In this programming assignment, you will implement the k-means algorithm to do clustering.

1. (10 points) Your executable should have the name "kmeans", and it takes arguments as follows:

   ./kmeans -database_file=database.txt -k=k -max_iters=n -eps=e -output_file=output.txt

   where database.txt (1 point) is the input file with a database of data points (its format will be discussed later), k (1 point) is the number of clusters you want to have, n (1 point) is the number of iterations you will allow in k-means, e (1 point) is the minimum distance you allow behind new and old centroids (will be discussed laster) and output.txt (1 point) is the output file into which your algorithm outputs the clusters (its format will be discussed later).

   You need to provide a README file in which you provide a command line (5 points) similar to the above one such that by copy-pasting your commend line on stdlinux, the TA should be able to run your algorithm. That also means, you have to make sure your algorithm runs on stdlinux.

2. (10 points) You need to implement a function named **read_data** (8 points) to read in the data points from the database file. You are free to use whatever data structures as you see fit to represent the data points. In your README file, you need to specify what data structures you use for the representations (2 points).

   Database file format: an example of the database file is as follows:

   0.4 0.5 0.1 0.2
   0.1 0.4 0.3 0.2
   0.1 0.9 0.1 0.4

   that is, each line of the file represents a data point, and each number in a line represents the value of the data point on a feature/attribute. In other words, the database is represented as a full matrix: each row of the matrix represents a data point, each column of the matrix represents a feature/attribute, each element in the matrix is the value of the corresponding data point on a corresponding feature/attribute.

3. (88 points) You need to have a function named **genKmeans** according to the following k-means algorithm pseudo code

```
1
2     def genKmeans(database, k, n, e output_file):
3
4     # 10 points
5     randomly initialize k centroids
6
7     # repeat untill the new and old cluster centroids are e-difference or
8     # many iterations already
9     Repeat until convergence w.r.t e (10 points) or n iterations (3 points)
10
11        # 40 points
12        assign each data point to each of the k clusters based on Euclidean distance
13
14        # 10 points
15        update the cluster centroids
16
17    # 15 points
18    output the k clusters
19
20    return
```

Output file format: the output file should have the following format:

0: 1 4 5 8 10 12
1: 2 3 6 7 9 13

that is, in each line, the first number is the cluster id (e.g., "0: " in the above example), the rest numbers are the data point ids that are clustered in the cluster. All the ids are in C-style indexing (i.e., starting from 0) and should be sorted in increasing order.

4. * Bonus credits (10 points): You will get bonus credits if you can minimize the number of times you calculate distances.

   What to submit:

   1. Your source code in a zipped file. You should name your zipped file as X_Y_PA3.tar.gz where X is your first name (capital letters) and Y is your last name (capital letters). Please following the naming schemes specified, otherwise, **20** points will be taken off. This is for the TA to quickly find your submission.
   2. A README file with required content as described earlier.
   3. It is mandatory that at least 1/4 of the lines (not including empty lines) in your code should be dedicated to comments. Otherwise, **20%** of the points that you will get will be taken off (non-genotiable!)
   4. You need to make sure your code runs on stdlinux. If your code does not run on stdlinux, you will get points based on your implementation of the required functions. However, you will lose 20% of the points that you can get from the implementation.
   5. You need to implement the algorithm as required and follow the given pseudo code. You will be graded only based on what you correctly implemented.