Homework1: Local space requirements analysis

This analysis aims to provide insight into the local memory requirements of this program, by providing a bound to the local space indicator M_L required by the function ***MRPrintStatistics***.

Local space M_L represents the maximum amount of local (or main) memory required by a single invocation of a map or reduce function, for storing the input and any data structure needed by the invocation (including all possible shared variables). The maximum is taken over all rounds and all invocations in each round.

All the transformations used in ***MRPrintStatistics***:

1. **Map 1:** We can denote the total number of input points as N, and consider all the points and centroids to have dimension d. Since we defined L as our number of partitions, assuming balanced partition, each worker should have approximately N/L points to process. Each point from the batch is mapped to its nearest centroid from the set of centroids C.
   During this transformation, each input point is stored in the local memory (d floats), along with the set of all centroids ($K \times d$ floats). Since points are processed one by one in *map()* (Spark doesn't keep all the input points in memory at once), the total amount of local memory required by a single map transformation on each partition is: $O(K \times d)$. Since *map()* returns a new RDD by applying the transformation function to each input point, at most N/L points and K centroids will be stored in local memory at once. That means: $O(N/L \times d + K \times d)$.

2. **Reduce 1 (*reduceByKey()*):** Points are grouped together by their keys which represent a combination of the nearest centroid and label (demographic group). We can have at most $K \times 2$ different groups since we have K centroids, and two possible different labels (A and B). After performing the reduce function, each output pair contains a key and a partial count per cluster, per demographic group. Local memory required by this transformation is at most: $O(K \times 2) = O(K)$.

3. **Map 2:** In the second map phase, we use a function that transforms the *((clusterIndex, label), count)* to *(clusterIndex, (label, count))* because that will allow us to later group these entries by *clusterIndex*. Local memory required by this simple reshape transformation is $O(K)$ per worker.

4. **Reduce 2 (*groupByKey()*):** In the second reduce phase, all entries are grouped by the *clusterIndex* key. In this case, we can have at most $K \times 2$ different keys on input on a single worker. That means that local space complexity per worker is at most: $O(K \times 2) = O(K)$.

5. **.*sortByKey().collect()*:** This does not impact workers' local space, instead, all the data is loaded into the driver's memory so that we can loop through them and print them out.

Finally, the local memory M_L required by a single partition is bounded by:

$$M_L = \max\{O(N/L \times d + K \times d), O(K), O(K), O(K)\}$$

$$M_L = O(N/L \times d + K \times d) = O(N/L + K)$$

assuming $N/L >> K$: $$\boldsymbol{M_L = O(N/L)}.$$

We can conclude that in this case the M_L is dependent of the total number of input points N (and the number of clusters K and the dimensionality d).