1. A short description of how you went about parallelizing the k-medoids algorithm. You should include how you decomposed the problem and why, i.e., what were the tasks being parallelized.

   I divided the problem into three main tasks. The first task is assigning each point to the nearest medoid, the second is updating the medoids, and the third is checking if any medoids have changed.

   I decomposed the tasks this way because assigning points to the closest medoids can be done in parallel by dividing the points across threads, allowing each thread to handle a subset of the points. Similarly, updating the medoids requires calculations for each cluster, so I distributed the clusters across threads to perform these calculations in parallel.

   In the second task, updating the medoids, I made it so that if there was even a single change in any of the medoids, it would return 1 to indicate a change. Thus, I checked it in the third task. If no changes occurred, the loop would break.

   Thus, I parallelized the processes of assigning points to their nearest medoids and updating the medoids.

2. Timing results for 1, 2, 4, 8, and 16 threads for k-medoids clustering.
   1. km_pthreads with 256 clusters and 1 thread
      - It took 30 minutes.

      km_openmp with 256 clusters and 1 thread
      - It took 25 minutes.
   2. km_pthreads with 256 clusters and 2 thread
      - It took 23 minutes.

      km_openmp with 256 clusters and 2 thread
      - It took 18 minutes.
   3. km_pthreads with 256 clusters and 4 thread
      - It took 12 minutes.

km_openmp with 256 clusters and 4 thread

- It took 9 minutes.

4. km_pthreads with 256 clusters and 8 thread

- It took 8 minutes.

Km_openmp with 256 clusters and 8 thread

- It took 7 minutes.

5. km_pthreads with 256 clusters and 16 thread

- It took 6 minutes.

Km_openmp with 256 clusters and 16 thread

- It took 5 minutes.

6. km_pthreads with 512 clusters and 1 thread

- It took 1 hour.

Km_openmp with 512 clusters and 1 thread

- It took 50 minutes.

7. km_pthreads with 512 clusters and 2 thread

- It took 21 minutes.

Km_openmp with 512 clusters and 2 thread

- It took 20 minutes.

8. km_pthreads with 512 clusters and 4 thread

- It took 15 minutes.

Km_openmp with 512 clusters and 4 thread

- It took 12 minutes.

9. km_pthreads with 512 clusters and 8 thread

- It took 10 minutes.

Km_openmp with 512 clusters and 8 thread

- It took 8 minutes.

10. km_pthreads with 512 clusters and 16 thread

- It took 7 minutes.

Km_openmp with 512 clusters and 16 thread

- It took 6 minutes.

11. km_pthreads with 1024 cluster and 16 thread

- It took 50 minutes.

Km_openmp with 1024 clusters and 16 thread

- It took 35 minutes.