Subject: Parallel Programming
Project name: K-Means Clustering

Team-Members

1. Shat Chakra Pawar Amgothu (71075197),
2. Xiaoshan Zhou (31740706),
3. Sujal Choudhary (71640579).

1. Project aim:

- Implement K-Means clustering in both **serial** and **parallel** (OpenMP) versions.
- Compare their **performance and scalability**.
- Demonstrate how **parallel computing** can optimize time-consuming tasks in clustering large datasets.

2. Problem Overview:

1. Randomly initializing *k* centroids.
2. Assigning each data point to the **nearest centroid**.
3. Recomputing centroids based on current assignments.
4. Repeating steps 2–3 until convergence or reaching max iterations.

In a **serial implementation**, all these steps run **sequentially**, which can be slow for large datasets.

In the **parallel implementation**:

- Distance calculations and centroid updates are done using **OpenMP**, enabling **multi-threading**.
- Each thread processes a subset of data points, significantly reducing runtime.

3. Formula:

$$d = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2 + (z_1 - z_0)^2}$$

4. Serial Version (Single-threaded)

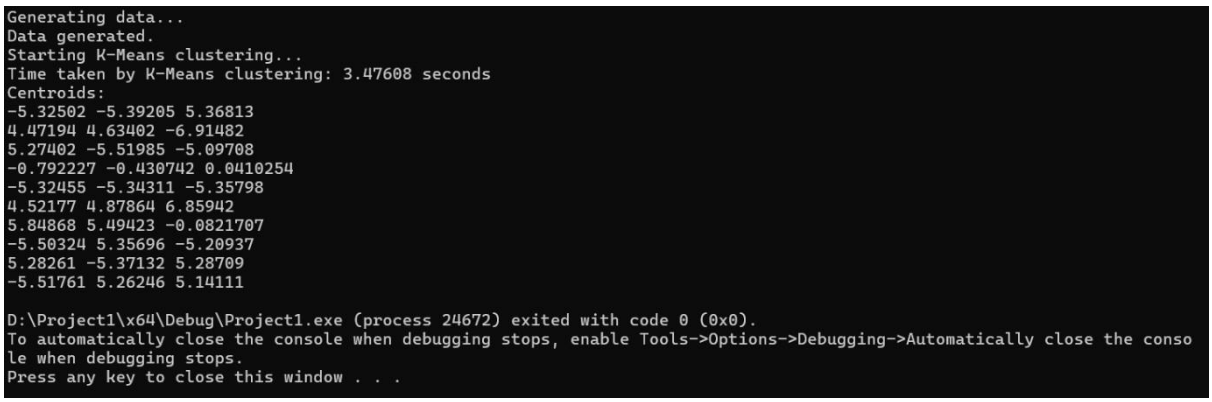
Purpose:

- To provide a **baseline** implementation of the algorithm.
- Ensures **correctness** and simplicity, it's easier to debug, read, and maintain.
- Useful for **small datasets** or when running on systems with limited processing power (e.g., microcontrollers or embedded systems).
- Helps in **comparing performance improvements** when moving to a parallel version.

5. Parallel Version (OpenMP / Multi-threaded)

Purpose:

- Designed to **speed up computation** by dividing work across multiple CPU cores using OpenMP.
- Especially beneficial when:
  - The dataset is **large**. ○ The number of **iterations or clusters** is high.
  - Real-time performance or **reduced execution time** is a requirement.
- Demonstrates the **scalability and efficiency** of the algorithm on modern hardware.



6. Why does parallelism matter?

1. K-Means involves repeatedly computing **distances** between every data point and every centroid

2. These operations become **increasingly expensive** as the number of Data Points (n) and clusters (k) increases.

3. To improve the runtime being dropped from **24.7s (serial & Single core)** to just **3.47s (parallel & multi core**

Total Threads	Run Time (Seconds)	Speed
1 (Serial)	**24.72**	**Single threaded**
2	**~12.5**	**2X speed**
4	**~6.4**	**4X speed**

6	**~4.9**	**Reduced time, but overhead chances may increase**
8	**~3.4**	**Best performance on the machine**