**Assignment 5 - Group 61**

Team Member: Aakash Chityala(2023005), Parsh Jain (2023368)

We implemented this by utilizing threads for this. It shows how variables can be captured in lambda expressions by value or reference and integrates these with multithreaded operations. The program implements two types of parallel loops: a 1D parallel\_for for handling tasks like a vector and a 2D parallel\_for for matrix-like computations. We also let user specify number of threads. This program show total execution time.

We used pthread library to implement this, using pthread\_create and pthread\_join API’s.

We implemented the parallel\_for functions by:

* **1D parallel\_for**: This function splits a 1D range [low, high) into smaller chunks, with each thread handling one chunk based on the total number of threads (numThreads). It uses a single\_thread\_args structure to pass details like the range and lambda function to the threads. Threads are created with pthread\_create, and their work is synchronized using pthread\_join to ensure proper execution.
* **2D parallel\_for**: This function expands parallel processing to a 2D range [low1, high1) x [low2, high2). Threads are assigned sub-ranges in both dimensions, using a matrix\_thread\_args structure to pass the necessary arguments. Each thread processes its assigned region independently before joining back.

Contributions:  
Aakash – Coding, Debugging,

Parsh – Coding, Debugging

<https://github.com/Parsh-Jain/OS_Assignment_5>