# Parallel Processing - 2025

Assignment 2 – MPI

## Deadline & Submission:

1. **Teams:** Max three students in the team.
2. Upload it on Classroom with file named: **A2_student1ID_student2ID_GroupName.zip**
   e.g., A2_20130001_20130002_S1_S2.zip
3. Code must be in C language, and MPI & you must run it before sending.
4. Attach a screen shot from the console output for each problem.
5. The team members must be different for each assignment.
6. Cheating could lead to serious consequences.
7. **Deadline is 10/5/2025 @11:59 PM.**

## Problem 1: Counting Primes

Write a parallel C program to count the prime numbers within an input range using the following two methods, then compare the execution times of both programs:

a) MPI_Bcast and MPI_Reduce ONLY
b) MPI_Send and MPI_Recv ONLY

**Given**

- Lower bound number x
- Upper bound number y

**Output**

- Count of prime numbers occurring between x and y.

### Parallelization Scenario:

**Master Process:**

- Calculate the subrange size **r = (y - x) / p** (if including master) or **(y - x) / (p - 1)** processes (without master).
- Broadcast **x** and **r** to each slave process using MPI_Bcast (or loop of MPI_Send).
- Receive sub-count from each slave process using MPI_Reduce (or loop of MPI_Recv).
- Print total count of primes between x and y.

**Slave Process:**

- Receive **x** and **r** through the MPI_Bcast call (or MPI_Recv).
- Calculate the lower bound **a,** and upper bound **b** according to its rank.
- Count primes in its subrange (between **a** and **b**).
- Send this partial count to the master process using the MPI_Reduce call (or MPI_Send).

## Example:

***n = 4, x = 1, y = 16     r = (16 - 1) / (4 - 1) = 5***
***p1: calculate partial count of prime numbers from 1 to 5       Count = 3 (2, 3, 5)***
***p2: calculate partial count of prime numbers from 6 to 10      Count = 1 (7)***
***p3: calculate partial count of prime numbers from 11 to 15     Count = 2 (11, 13)***
***After reduction, P0 will have Count = 6 (2, 3, 5, 7, 11, 13)***
***Note:** The length of the range may not be divisible by the number of processes. So, you should handle this case.*

## Problem 2: Matrix Summation

Write a parallel C program using MPI to compute the sum of two matrices A and B and store the result in matrix C, using the following MPI functions:

- MPI_Scatter
- MPI_Gather

**Given:**

- Two matrices A and B of size N x M (input from user or generated).
- Each process will compute a portion of the result matrix C = A + B.

**Output:**

- The final matrix **C** printed by the **master process** (process 0).

## Parallelization Scenario:

**Master Process:**

- Create and initialize matrices **A** and **B** (random values or user input).
- Flatten them to 1D arrays for communication.
- Use MPI_Scatter to distribute equal chunks (rows or elements) of both **A** and **B** to all processes.
- Use MPI_Gather to collect the computed chunks of matrix **C** from all processes.
- Reconstruct and print the final matrix **C**.

**Slave Processes:**

- Receive their chunk of data for **A** and **B** using MPI_Scatter.
- Perform element-wise addition of their chunk: **C = A + B**.
- Send the result back to the master using MPI_Gather.

**Note:** The number of rows **N** should be divisible by the number of processes (or handled accordingly if not).