MPI Practice

Write the proper code to solve the following problems. Show screenshots of the output.

---

1. Hello World with MPI

- **Problem Statement:**

Write an MPI program where each process prints a message that includes its rank and the total number of processes. The output should look like:

"Hello from process X out of Y processes."

---

2. Sum of Array Elements (Distributed Summation)

- **Problem Statement:**

Create an MPI program that splits an integer array among multiple processes. Each process computes the sum of its portion and sends it to the root process. The root process then calculates and displays the total sum of the array.

Use MPI_Scatter to distribute the array and MPI_Reduce to collect partial sums.

---

3. Broadcast a Number

- **Problem Statement:**

The root process (rank 0) accepts an integer input (either hardcoded or read from user input). This number is then broadcasted to all other processes using MPI_Bcast. Every process prints the number it receives.

---

4. Find Maximum Number

- **Problem Statement:**

Each process starts with a unique integer value (e.g., its rank multiplied by 3). Write an MPI program that uses MPI_Reduce to find the maximum number among all processes and prints it from the root process.

---

5. Ring Communication

- **Problem Statement:**

Implement a ring communication pattern where each process sends a number to its right neighbor (modulo total number of processes) and receives a number from its left neighbor. Each process prints the value it received.

Use MPI_Send and MPI_Recv for communication.

6. Prefix Sum (Scan)

- **Problem Statement:**

Write an MPI program where each process starts with a local integer value. Use MPI_Scan to calculate the **prefix sum** across processes. Each process should print its own prefix sum result.

---

7. Scatter and Gather Strings

8. Matrix Row Distribution

- **Problem Statement:**

Given a 2D matrix (initialized at the root), distribute one row to each process using MPI_Scatter. Each process computes the sum of the elements in its row. Use MPI_Gather to send these row sums back to the root process for display.

Assume the number of rows equals the number of processes.

---

9. Barrier Synchronization with Execution Time Measurement

- **Problem Statement:**

Write an MPI program where each process prints a message **before** and **after** a synchronization point. Use MPI_Barrier to ensure that all processes reach the barrier before continuing.

In addition, measure and display the **execution time** taken by each process (from the start of the program to just after the barrier). This will help demonstrate the synchronization effect of MPI_Barrier and how timing differs across processes.

- **Requirements:**

- Use MPI_Wtime() to measure wall-clock time.
- Print each process's execution time individually.
- Clearly indicate when each process enters and exits the barrier.

- **Expected Output Example:**

```
Process 1: Before barrier
Process 0: Before barrier
Process 2: Before barrier
Process 0: After barrier | Execution Time: 0.003124 seconds
Process 1: After barrier | Execution Time: 0.004233 seconds
Process 2: After barrier | Execution Time: 0.004567 seconds
```

10. Check Even or Odd Rank

- **Problem Statement:**

Each process checks if its rank is even or odd and prints a corresponding message. For example:

"Process 2 is even"

"Process 3 is odd"

Due Date: 15/4/2025 at 11:30pm via blackboard.
Submit a single PDF file containing your answers.