LAB ASSIGNMENT- 1

Exercise 1: Basic Thread Parallelism

Objective: Initialize the OpenMP environment and verify multithreaded execution.

- **Task:** Write a C/C++ program that uses the #pragma omp parallel directive to print "Hello World."
- **Key Concept:** Understand the "Fork-Join" model where the master thread spawns a team of worker threads.

Exercise 2: Runtime Configuration and Data Environment

Objective: Learn to pass parameters to the parallel region at runtime and use OpenMP clauses.

- **Task:** Write a program that:
    1. Prompts the user to enter the number of threads at runtime.
    2. Uses the num_threads() clause to set the thread size.
    3. Prints: "Hello World from Thread [ID] of [Total Threads]."

Exercise 3: Analyzing Race Conditions and Solution

Objective: Observe the non-deterministic nature of parallel execution and the impact of shared resources.

- **Task:** Write an OpenMP program to print "Hello World" do not ma
- **Requirement:** Introduce a manual delay (using a "wait" condition or a heavy dummy loop) inside the parallel block to force threads to overlap.
- **Observation:** Document how the output becomes scrambled (interleaved) due to multiple threads accessing the standard output simultaneously, demonstrating a **Race Condition**.
- **Solution:** Use the private() clause to ensure each thread has its own copy of the thread ID variable.

Exercise 4: Large-Scale Vector Addition (Data Parallelism)

Objective: Implement data decomposition on large arrays to measure parallel efficiency.

- Task: 1. Create three 1D arrays of size 1,000,000.

2. Sequentially: Initialize two arrays with random or incremental values.

3. Parallely: Use #pragma omp parallel for to calculate the sum: C[i] = A[i] + B[i].

- **Requirement:** Compare the performance of the parallel loop against a standard sequential loop.

Exercise 5: Parallel Reduction for Maximum Value

Objective: Use OpenMP synchronization or reduction clauses to find maximum value in a dataset.

- Task: 1. Insert values in a 1D array of size 1,000,000 with random integers sequentially.

2. Parallely: Find the maximum value in the array.

- **Requirement:** Students should implement this using the reduction(max: variable) clause to avoid manual locking while ensuring thread safety.