**Class:** Third Year B.Tech(Computer Science and Engineering)

**Year:** 2025-26 **Semester:** Odd

**Course:** Cutting Edge Technologies Lab

**Course code:** 7CS352

**Practical No. 3**

**Exam Seat No:**

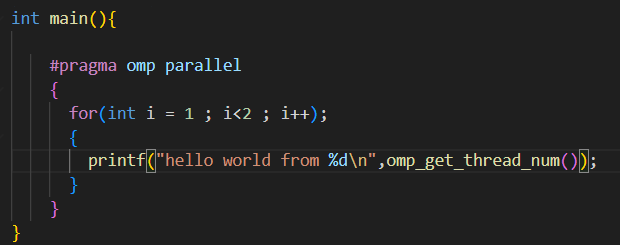
**Title of practical:**

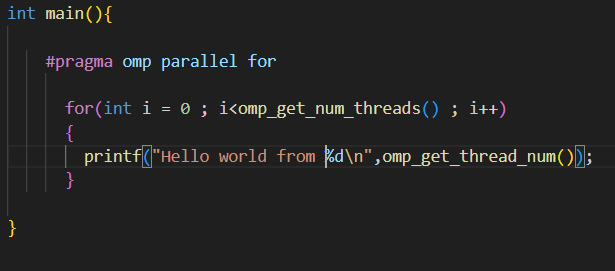
Parallelizing loops with #pragma omp parallel for, and applying scheduling clauses

1. Print hello world using, #pragma omp parallel and #pragma omp parallel for
2. Addition of Scalar value with a vector using static scheduling
3. Addition of Scalar value with a vector using Dynamic scheduling scheduling

**Problem Statement 1:**

**Screenshots:**

****

****

**Output ;**

**hello world from 0**

**hello world from 4**

**hello world from 5**

**hello world from 1**

**hello world from 9**

**hello world from 7**

**hello world from 3**

**hello world from 8**

**hello world from 6**

**hello world from 2**

**hello world from 11**

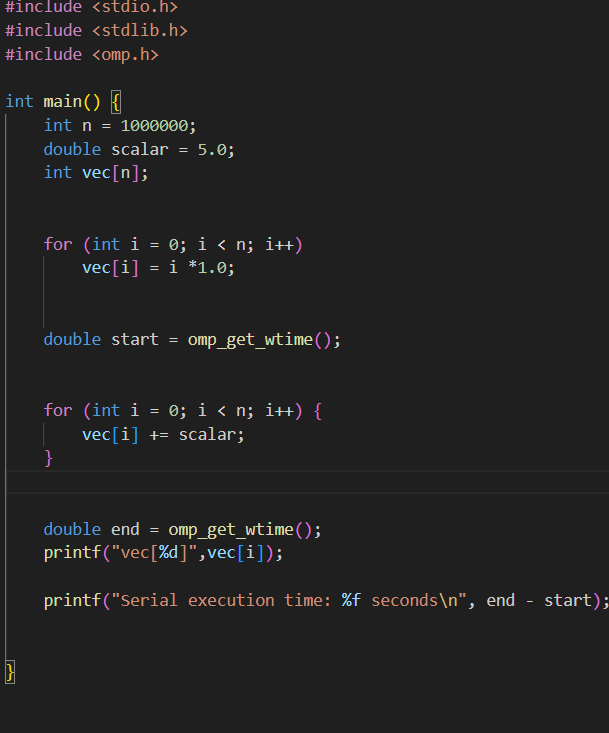
**hello world from 10**

**Information and analysis: Tima analysis, Speedup analysis for all programs**

* **Speedup Result: No Significant Speedup**
* **#pragma omp parallel: Creates a team of threads, and each thread executes the entire loop independently. This results in redundant work (16 print calls for 4 iterations).**
* **#pragma omp parallel for: This is a work-sharing construct that divides the loop iterations among the threads. Each iteration is executed only once by one thread, which is the correct way to parallelize a loop.**

**Problem Statement 2:**

**Screenshots:**

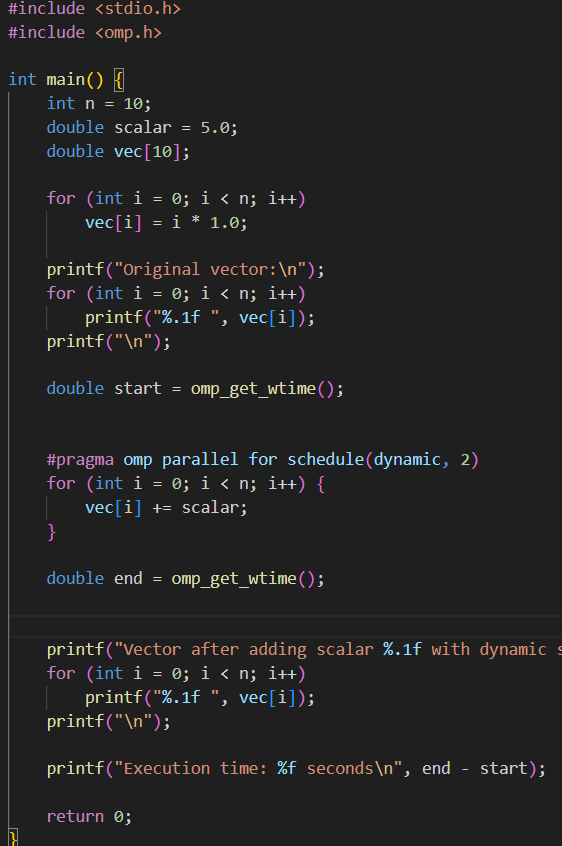
****

**Information and analysis:**

**Static execution time: 0.002151 seconds**

**Problem Statement 3:**

**Screenshots:**

****

**Information and analysis:**

**Execution time: 0.000680 seconds**

**Github Link:**