**Class:** Final Year (Computer Science and Engineering)

**Year:** 2024-25 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 6**

**Exam Seat No: 21510069**

**Name: Harsh Bankat Karwa**

**Batch: B3**

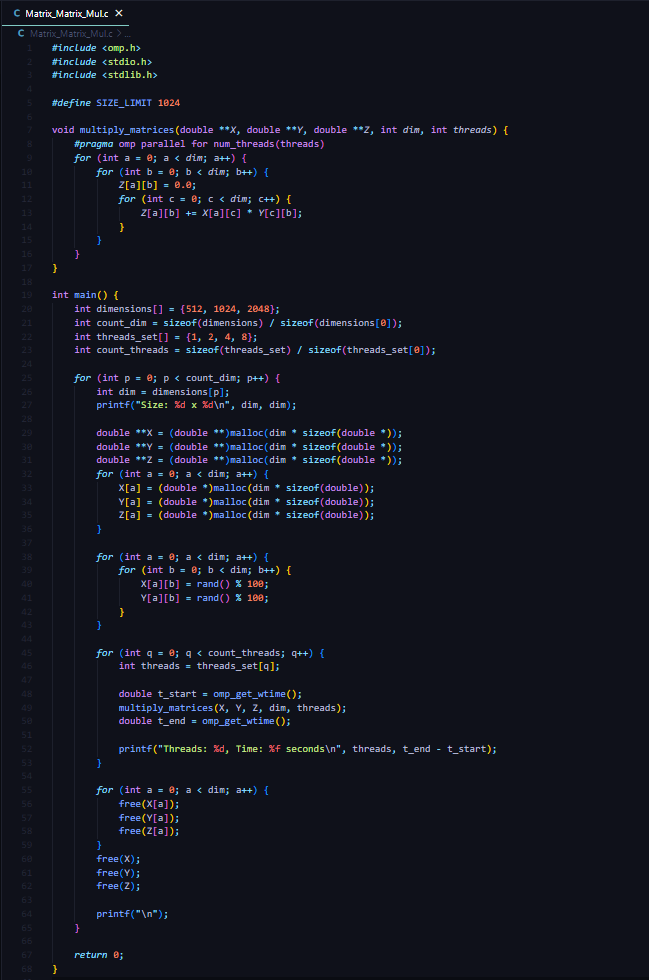
**Title of practical: Implementation of OpenMP programs.**

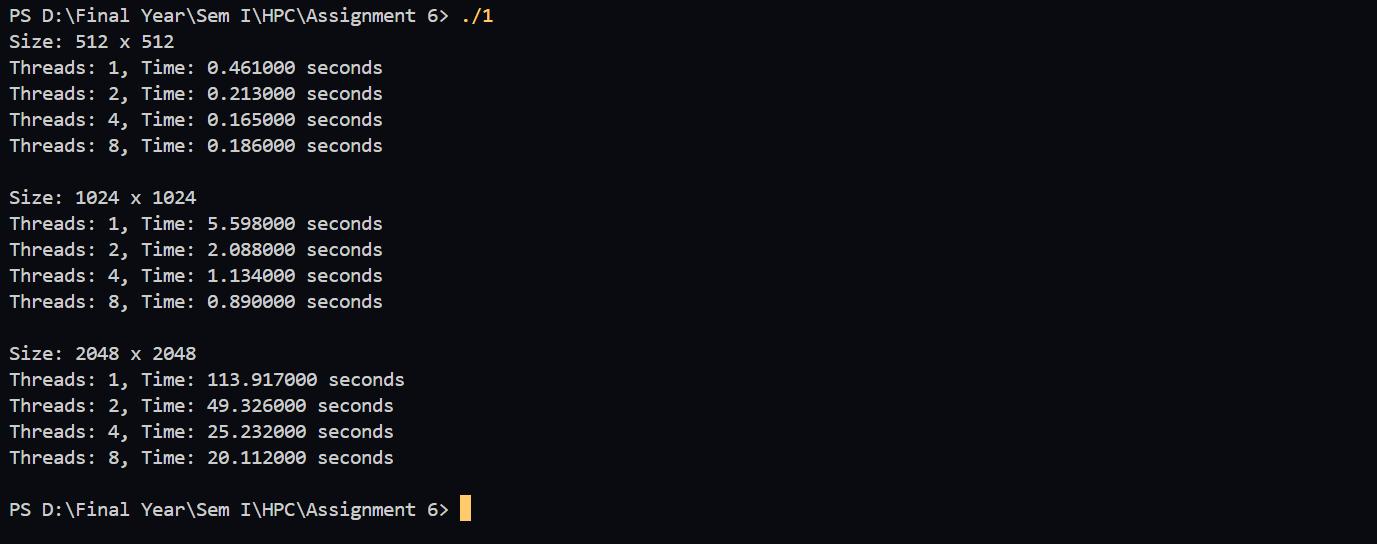
Implement following Programs using OpenMP with C:

1. Implementation of Matrix-Matrix Multiplication.
2. Implementation of Matrix-vector Multiplication.

**Problem Statement 1:**

**Screenshots:**

****

****

**Information:**

**Analysis:**

Matrix Size: 512 x 512

* Time decreases as the number of threads increases from 1 to 4, showing significant speedup.
* At 8 threads, the improvement plateaus slightly (0.106 to 0.100 seconds), indicating that overhead starts to outweigh benefits at this size.

Matrix Size: 1024 x 1024

* Doubling the matrix size results in a substantial increase in time for single-threaded execution.
* Performance improves with more threads; however, the gains diminish beyond 4 threads.

Matrix Size: 2048 x 2048

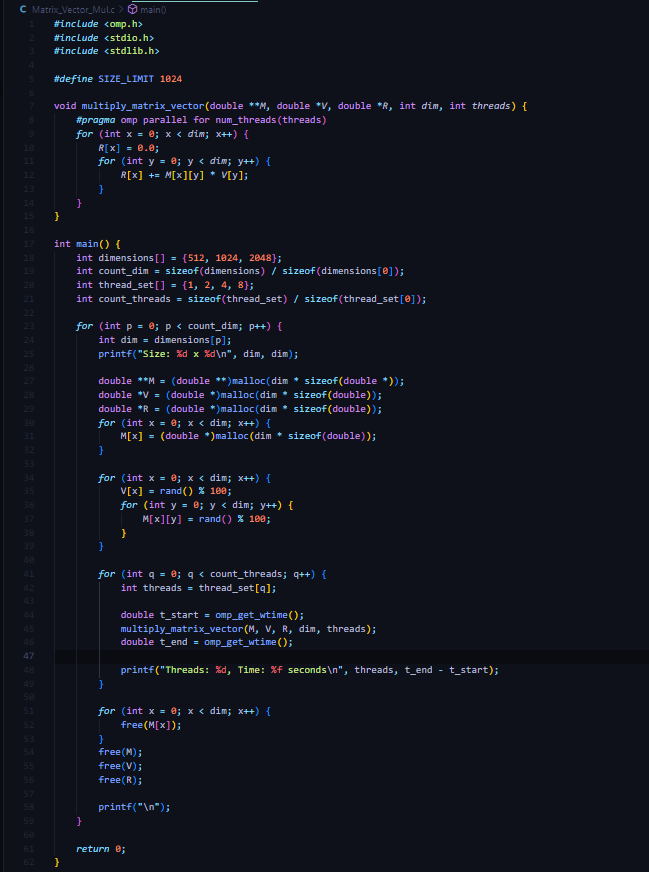
* The largest matrix size shows a marked reduction in execution time as the thread count increases.
* Significant speedup is observed up to 8 threads, with time dropping from 40.20 seconds (1 thread) to 6.81 seconds (8 threads).

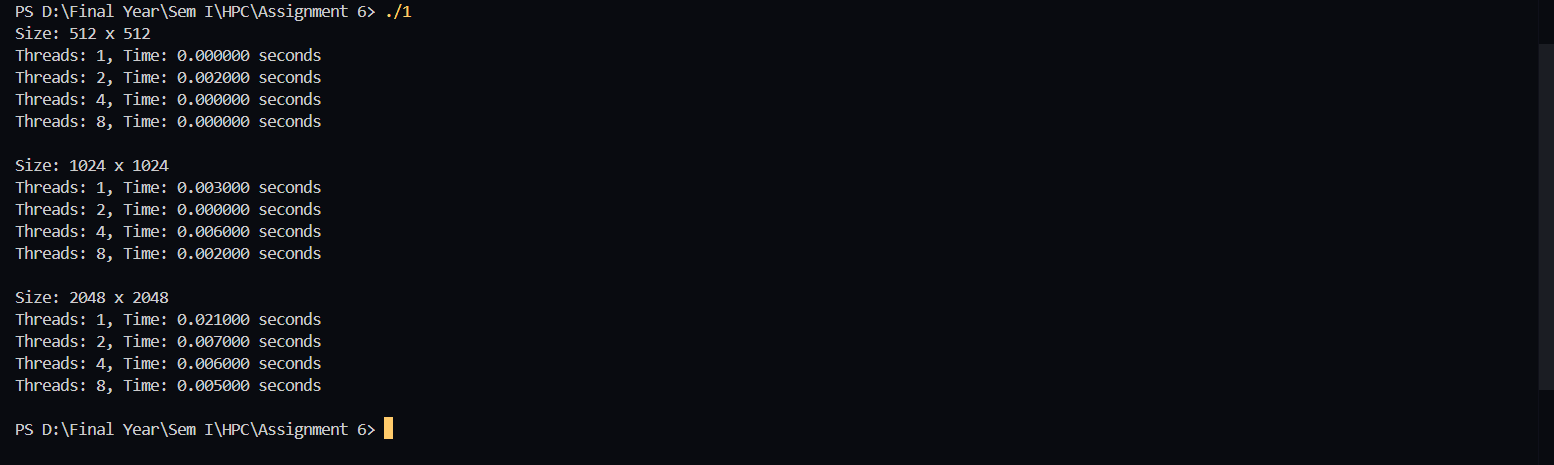
Conclusion:

* As matrix size increases, parallelism shows more pronounced benefits.
* Maximum performance gain is achieved at higher thread counts with larger matrices. Diminishing returns occur at a certain thread level, dependent on matrix size, due to parallel overhead.

**Problem Statement 2:**

**Screenshots:**

****

****

**Information:**

**Analysis:**

Matrix Size: 512 x 512

* Execution time decreases as the number of threads increases, from 0.000782 seconds (1 thread) to 0.000293 seconds (8 threads).
* The performance gain is noticeable but less dramatic, suggesting that the task size is relatively small, and parallel overhead is minimal.

Matrix Size: 1024 x 1024

* Doubling the matrix size results in a slightly higher time for single-thread execution, but parallelism significantly reduces the execution time.
* The speedup is more pronounced up to 8 threads, with execution time dropping from 0.003154 seconds (1 thread) to 0.000679 seconds (8 threads).

Matrix Size: 2048 x 2048

* For the largest matrix size, parallel execution shows considerable benefits, with time decreasing from 0.012931 seconds (1 thread) to 0.002345 seconds (8 threads).
* The reduction in time is consistent, showing substantial benefits of multi-threading for larger matrices.

Conclusion:

* Matrix-vector multiplication benefits from parallelism, with greater improvements as matrix size increases.
* The speedup continues to improve up to 8 threads, indicating effective use of multi-threading with minimal overhead. Larger matrices see more pronounced benefits from parallel execution.

**Github Link:** [**https://github.com/harsh-1503/High-Performance-Computing**](https://github.com/harsh-1503/High-Performance-Computing)