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

**Year:** 2025-26 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 5**

**PRN:22510092**

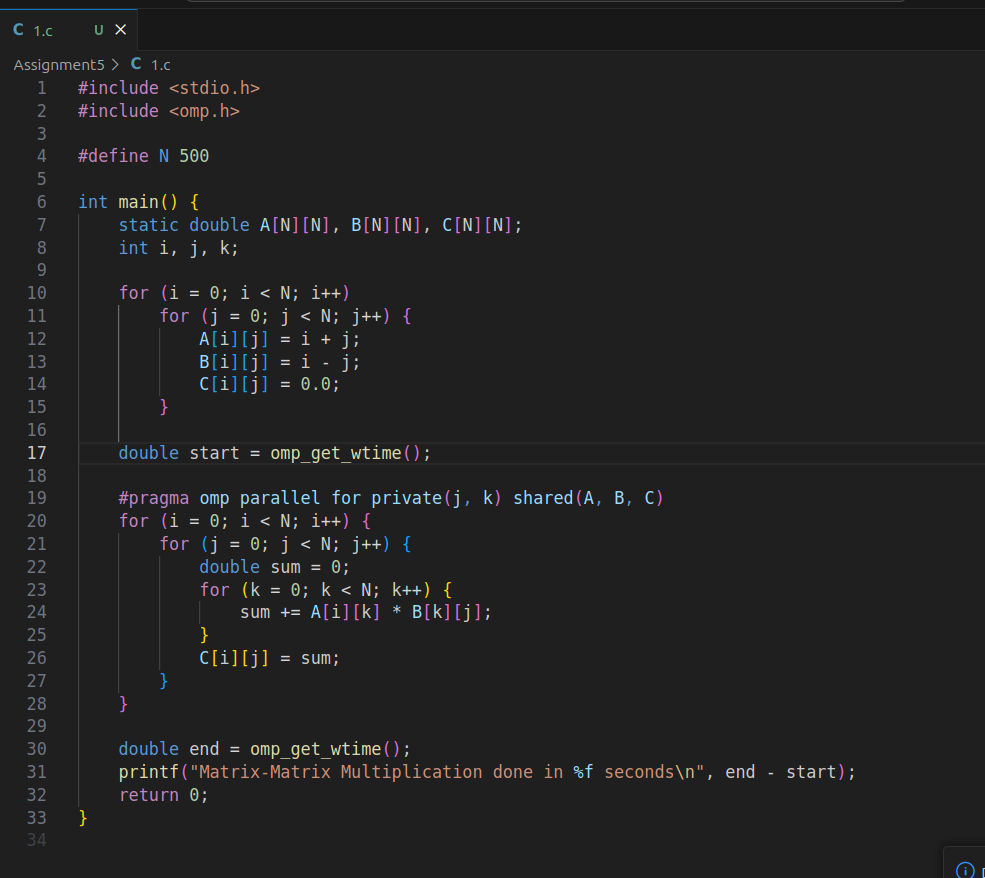
**Title of practical: Implementation of OpenMP programs.**

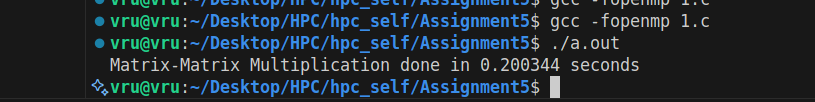
Implement following Programs using OpenMP with C:

1. Implementation of Matrix-Matrix Multiplication.
2. Implementation of Matrix-scalar Multiplication.
3. Implementation of Matrix-Vector Multiplication.
4. Implementation of Prefix sum.

**Problem Statement 1:** Implementation of Matrix-Matrix Multiplication.

**Screenshots:**

****



### ****Information:****

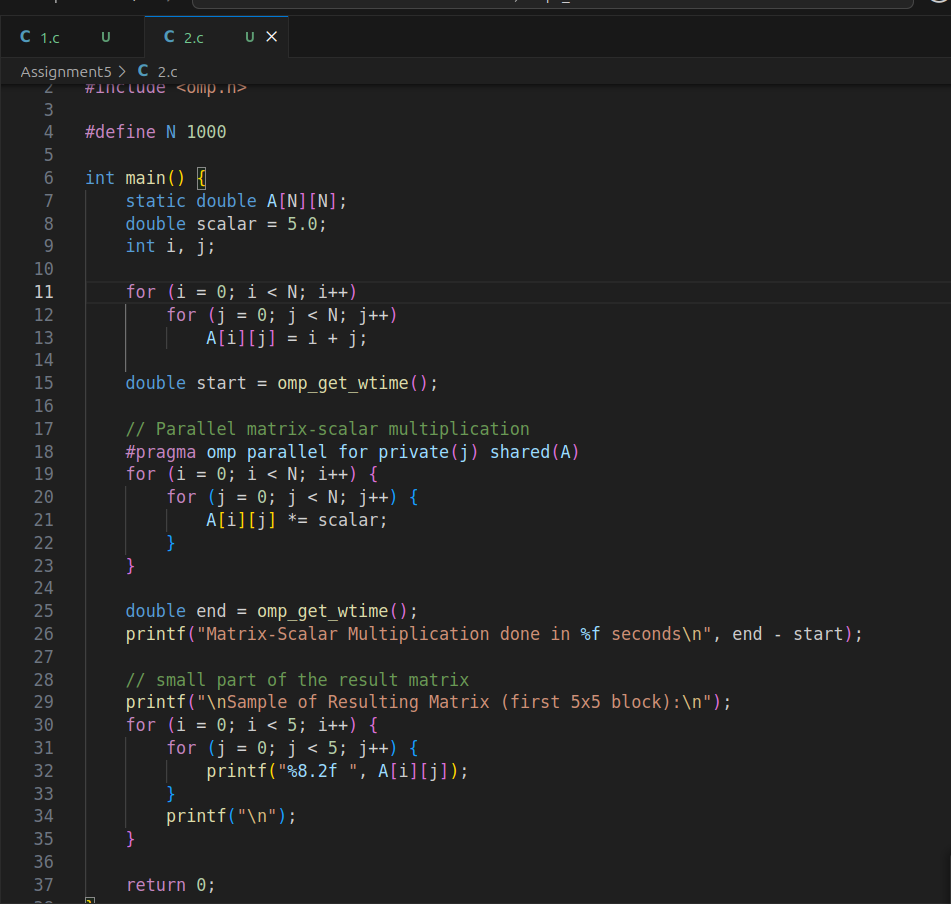
* This program multiplies two matrices A and B using **parallel for loop with OpenMP**.
* #pragma omp parallel for distributes the work among multiple threads.

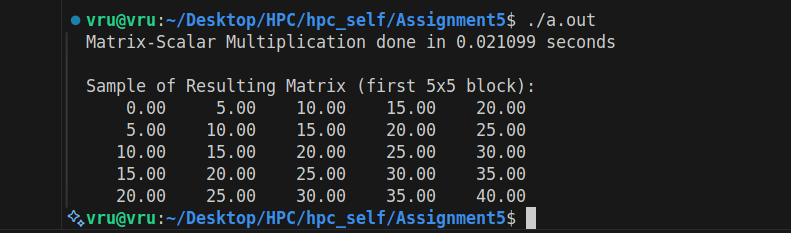
### Analysis:

* Sequential matrix multiplication is **O(N³)**.
* With OpenMP, tasks are divided across threads → speedup depends on number of cores.
* Ideal for large matrices.

**Problem Statement 2:** Implementation of Matrix-scalar Multiplication.

**Screenshots:**

****

****

**Information:**

Multiplies each element of a matrix by a scalar using OpenMP.

Parallel for improves speed by splitting element-wise operations across threads.

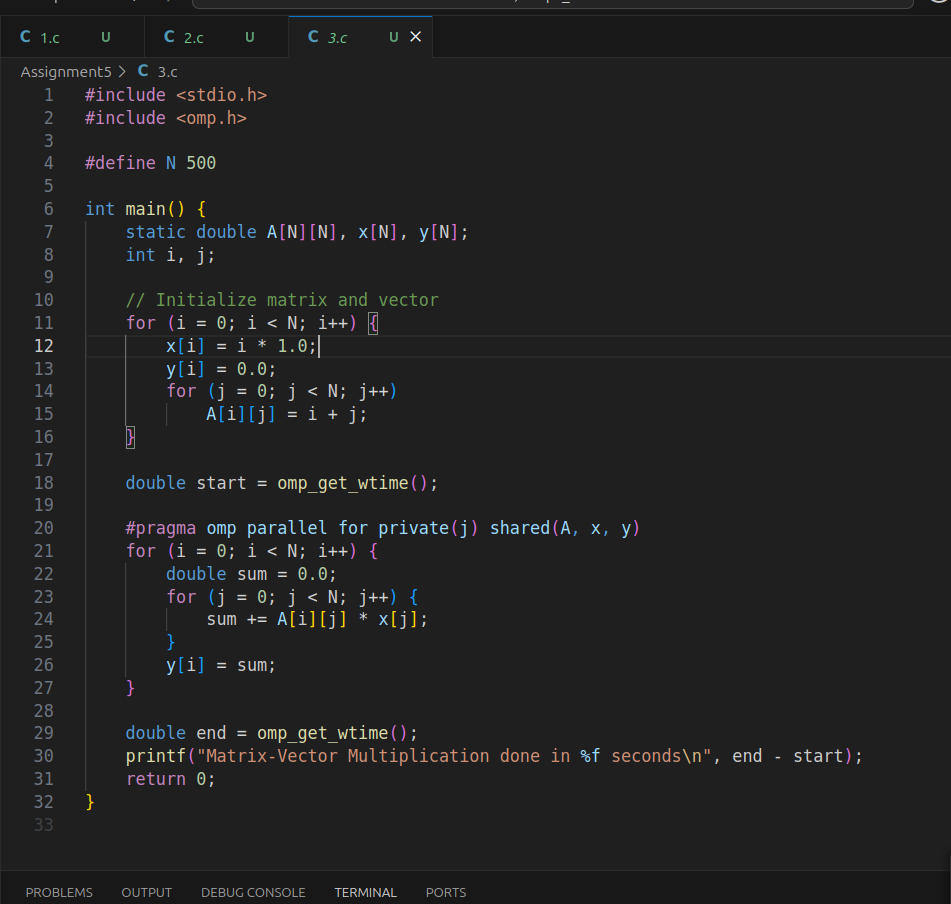
**Analysis:**

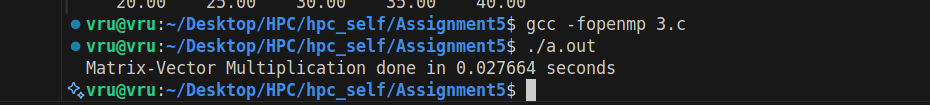
Very parallelizable because each element operation is independent.

Speedup depends on matrix size and thread count.

**Problem Statement 3:** Implementation of Matrix-Vector Multiplication.

**Screenshots:**



****

**Information:**

Computes y = A \* x where A is a matrix and x is a vector.

Uses row-wise parallelism for efficiency.

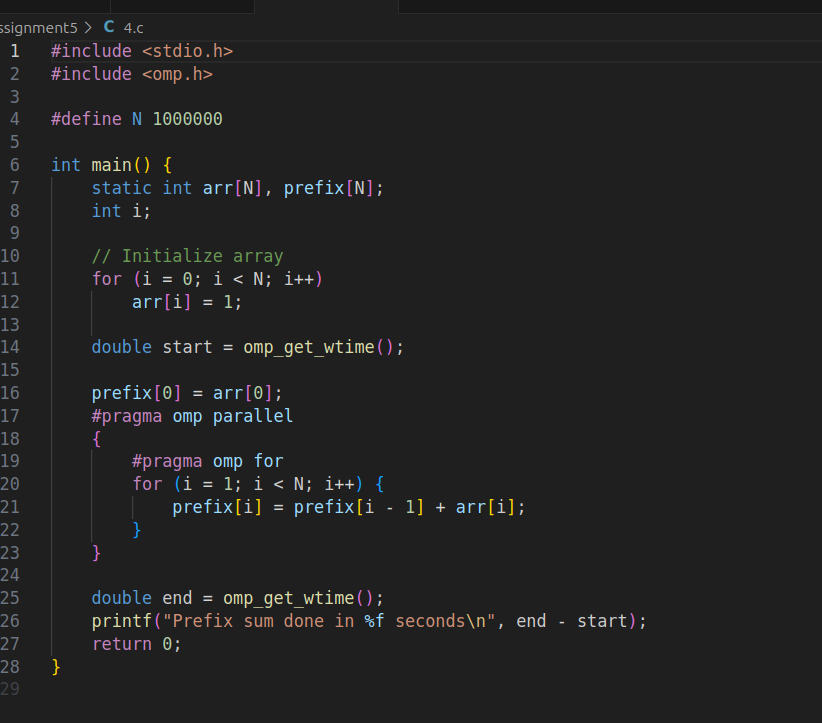
**Analysis:**

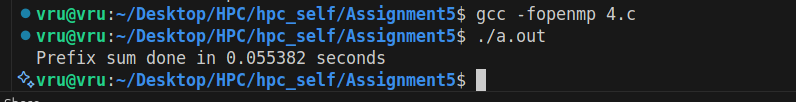
Ideal for parallelism because each row calculation is independent.

Performance gain depends on cache utilization and thread count.

**Problem Statement 4:**Implementation of Prefix sum.

**Screenshots:**

****



**Information:**

Prefix sum computes cumulative sums of an array.

Naive version shown; true parallel version requires a scan algorithm.

**Analysis:**

Straightforward serially but tricky for parallelization due to dependencies.

Parallel scan algorithms can achieve O(log N) depth.

**Github Link:**

**https://github.com/Vru01/HPC\_22510092**