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

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

**Course:** High Performance Computing Lab

**Practical No. 6**

**Exam Seat No:**

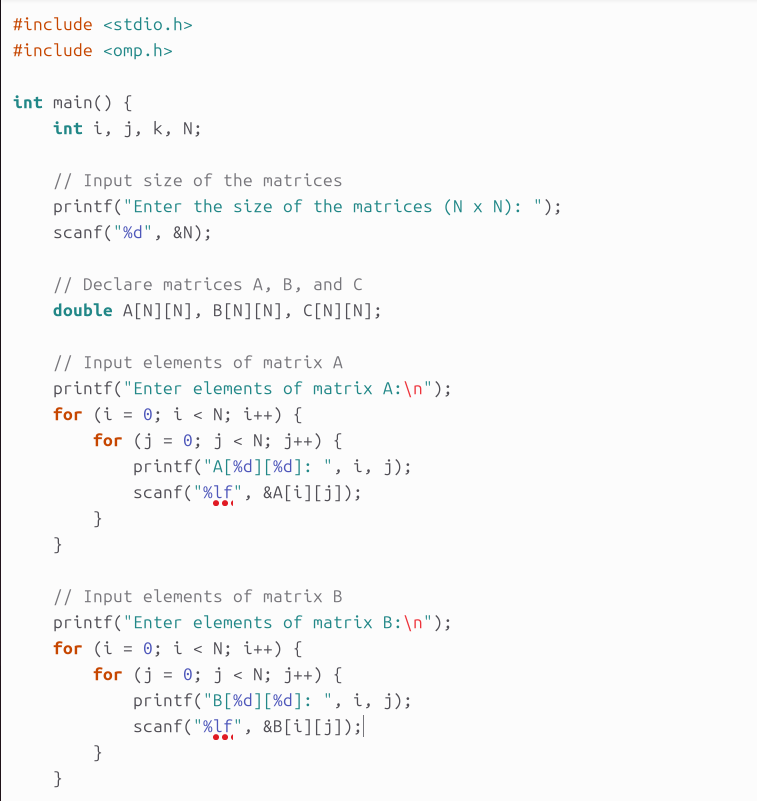
**Title of practical: Implementation of OpenMP programs.**

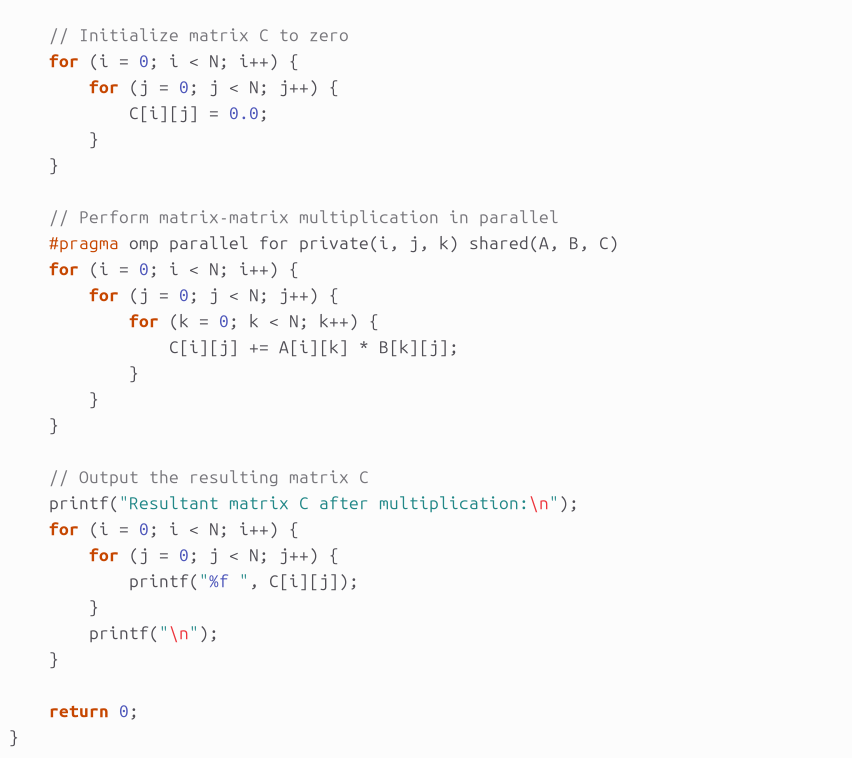
Implement following Programs using OpenMP with C:

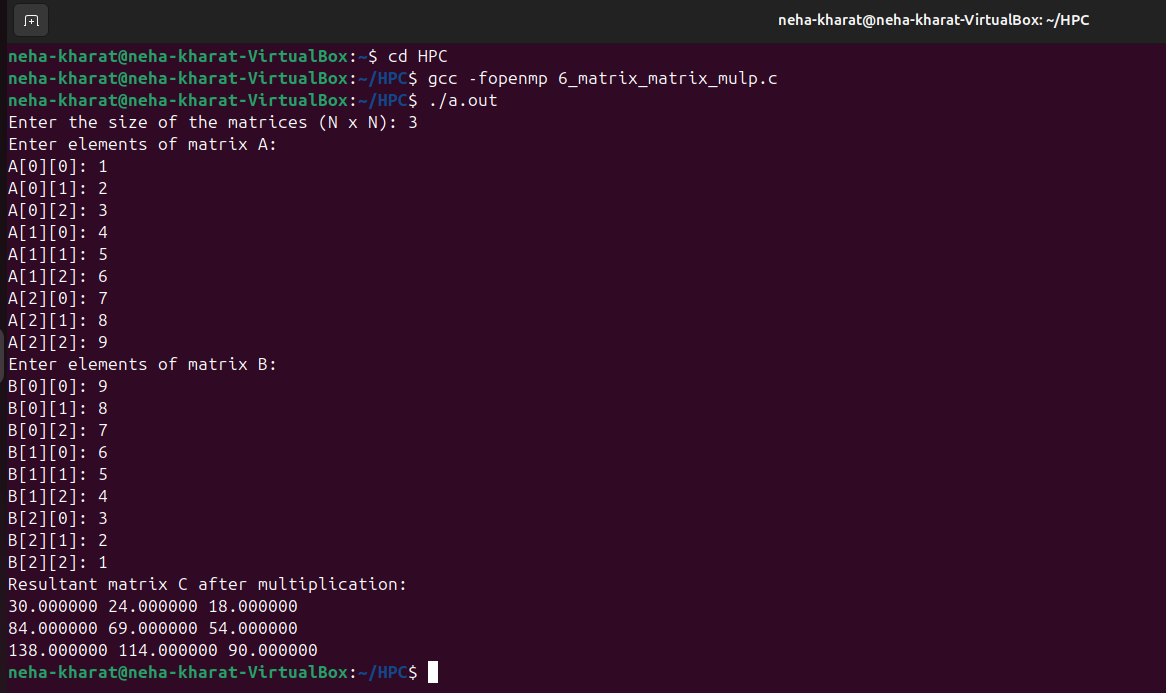
1. Implementation of Matrix-Matrix Multiplication.
2. Implementation of Matrix-vector Multiplication.

**Problem Statement 1:**

**Screenshots:**

****

****

****

**Information:**

 OpenMP Parallelization: The #pragma omp parallel for directive is used to parallelize the outer loop, which handles row-wise operations.

 Private and Shared Variables:

* i, j, k are declared as private to ensure each thread gets its own copy.
* A, B, C are shared as all threads need access to these matrices.

**Analysis:**

**Sequential Time Complexity:**

In this program, matrix-matrix multiplication involves three nested loops:

* The outer loop iterates over the rows of matrix A (i runs from 0 to N-1).
* The middle loop iterates over the columns of matrix B (j runs from 0 to N-1).
* The inner loop iterates over the elements in the row of matrix A and the column of matrix B to compute the dot product (k runs from 0 to N-1).

Thus, the time complexity of the sequential algorithm is: O(N^3)

where N is the number of rows/columns of the matrices.

**Parallel Time Complexity (with OpenMP):**

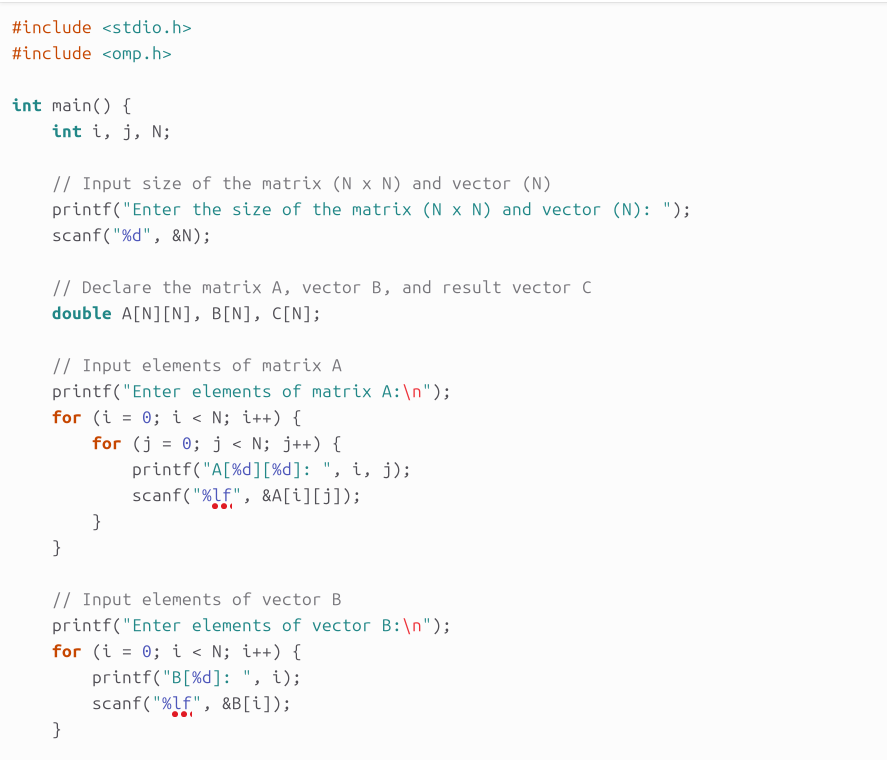
With OpenMP parallelization, the outer loop (over i, the rows of matrix A) is parallelized. Assuming perfect load balancing and no overhead, the number of operations can be evenly distributed across P threads, reducing the time complexity by a factor of P (the number of threads).

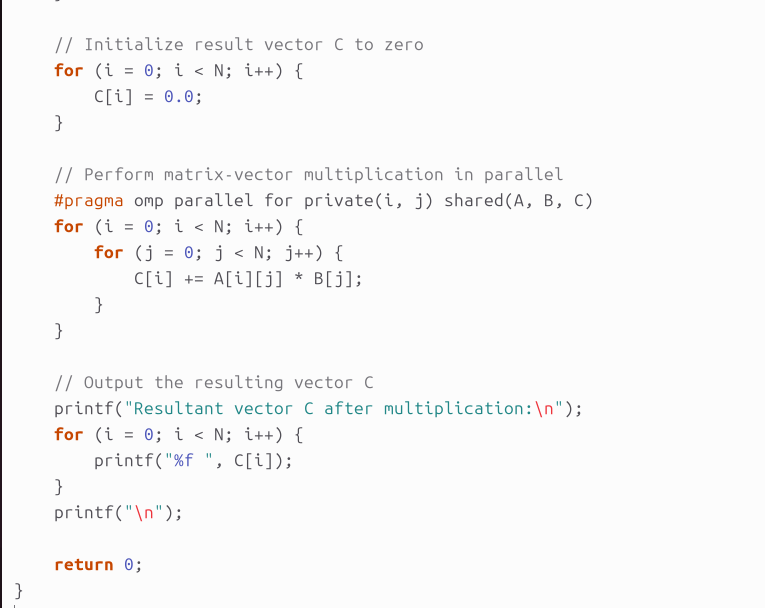
In the best-case scenario (ideal parallelism): O(N^3/P)

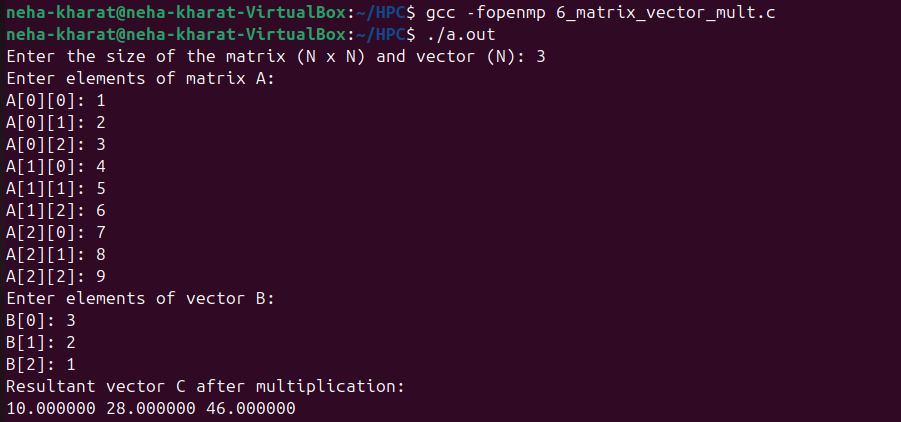
However, due to overhead from thread management, synchronization, and potential load imbalance, the actual performance may vary.

**Problem Statement 2:**

**Screenshots:**

****

****

****

**Information:**

**Analysis:**

**For matrix-vector multiplication:**

* **Sequential Time Complexity:**The computation involves multiplying each element of the matrix with the corresponding element of the vector and summing the results for each row. This results in: O(N^2)

where N is the number of rows/columns of the matrix.

* **Parallel Time Complexity (with OpenMP):**The matrix is divided among multiple threads, ideally reducing the time complexity depending on the number of threads P. In the best case scenario, with perfect parallelism: O(N^2/P)

**However, actual performance depends on factors like overhead from thread creation and data sharing, so the ideal speedup may not always be fully achieved.**

**Github Link:**

**https://github.com/NehaK1311/HPC/blob/main/Assign6/6\_matrix\_matrix\_mult.c**