Nagar Yuwak Shikshan Sanstha's
Yeshwantrao Chavan College of Engineering
(An Autonomous Institution affiliated to Rashtrasant Tukadoji Maharaj Nagpur University)
Hingna Road, Wanadongri, Nagpur - 441 110
NAAC A++
Ph.: 07104-237919, 234623, 329249, 329250 Fax: 07104-232376, Website: www.ycce.edu

Department of Computer Technology

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session 2025-2026

Vision: To help businesses uncover crucial insights	**Mission:** To be a good data scientist

Program Educational Objectives of the program (PEO): (broad statements that describe the professional and career accomplishments)

PEO1	**Preparation**	**P: Preparation**	**Pep-CL abbreviation pronounce as Pep-si-lL easy to recall**
PEO2	**Core Competence**	**E: Environment (Learning Environment)**	
PEO3	**Breadth**	**P: Professionalism**	
PEO4	**Professionalism**	**C: Core Competence**	
PEO5	**Learning Environment**	**L: Breadth (Learning in diverse areas)**	

Program Outcomes (PO): 1. Understand and Apply Parallel Programming Concepts

2. Analyse and Improve Program Performance.

3. Demonstrate Practical Skills in HPC Tools and Environments.

Keywords of POs:
Engineering knowledge, Problem analysis, Design/development of solutions, Conduct Investigations of Complex Problems, Engineering Tool Usage, The Engineer and The World, Ethics, Individual and Collaborative Team work, Communication, Project Management and Finance, Life-Long Learning

PSO Keywords: Cutting edge technologies, Research

"I am an engineer, and I know how to apply engineering knowledge to investigate, analyse and design solutions to complex problems using tools for entire world following all ethics in a collaborative way with proper management skills throughout my life." *to contribute to the development of cutting-edge technologies and Research*.

Integrity: I will adhere to the Laboratory Code of Conduct and ethics in its entirety.

Name and Signature of Student and Date
Soham pimpalgaonkar – 28/10/2025

Vision of the Department

To be a well-known centre for pursuing computer education through innovative pedagogy, value-based education and industry collaboration.

Mission of the Department

To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

Session	**2025-26 (ODD)**	**Course Name**	**HPC Lab**
Semester	**7**	**Course Code**	22ADS706
Roll No	62	**Name of Student**	Soham pimpalgaonkar

Practical Number	4
Course Outcome	1. Understand and Apply Parallel Programming Concepts 2. Analyse and Improve Program Performance
Aim	Matrix Multiplication using OpenMP
Problem Definition	Matrix Multiplication using OpenMP
Theory (100 words)	**OpenMP** (Open Multi-Processing) is an API (Application Programming Interface) that supports multiprocessing in shared-memory platforms. Applying OpenMP to matrix multiplication involves using directives to parallelize the computationally intensive loops, significantly reducing the execution time for large matrices. Core Concept: Parallelization of Matrix Multiplication Matrix multiplication of two matrices, (size) and (size), to produce matrix (size) is defined by: $$C_{ij} = \sum_{l=1}^{k} A_{il} B_{lj}$$ The standard serial algorithm uses three nested loops: 1. Outer loop (i): Iterates over the rows of (and). 2. Middle loop (j): Iterates over the columns of (and). 3. Inner loop (l): Calculates the dot product for . **Steps for execution:** Step 1: Write the serial (single-threaded) matrix multiplication code. Step 2: Compile and run the serial program Step 3: Add OpenMP parallelization and timing

<table>
<tr><td></td><td>Step 4: Compile and run the OpenMP version
Step 5: Compare results</td></tr>
<tr><td>Code:</td><td>

1. Matmul_serial.c

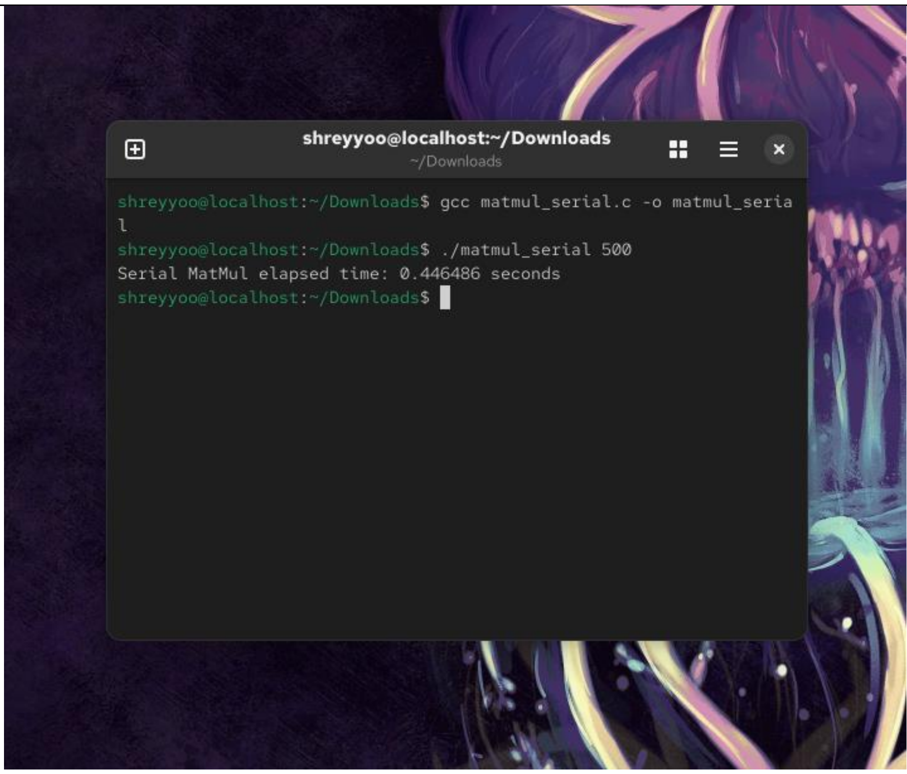
```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

void matmul(int N, double *A, double *B, double *C) {
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            double sum = 0;
            for (int k = 0; k < N; k++)
                sum += A[i*N+k] * B[k*N+j];
            C[i*N+j] = sum;
        }
}

int main(int argc, char **argv) {
    if (argc < 2) {
        printf("Usage: %s matrix_size\n", argv[0]);
        return 1;
    }
    int N = atoi(argv[1]);
    double *A = malloc(N*N*sizeof(double));
    double *B = malloc(N*N*sizeof(double));
    double *C = malloc(N*N*sizeof(double));

    // Initialize matrices A and B
    for (int i = 0; i < N*N; i++) {
        A[i] = 1.0;
        B[i] = 2.0;
    }

    clock_t start = clock();
    matmul(N, A, B, C);
    clock_t end = clock();

    double time_spent = (double)(end - start) / CLOCKS_PER_SEC;
    printf("Serial MatMul elapsed time: %f seconds\n", time_spent);

    free(A); free(B); free(C);
    return 0;
}
```

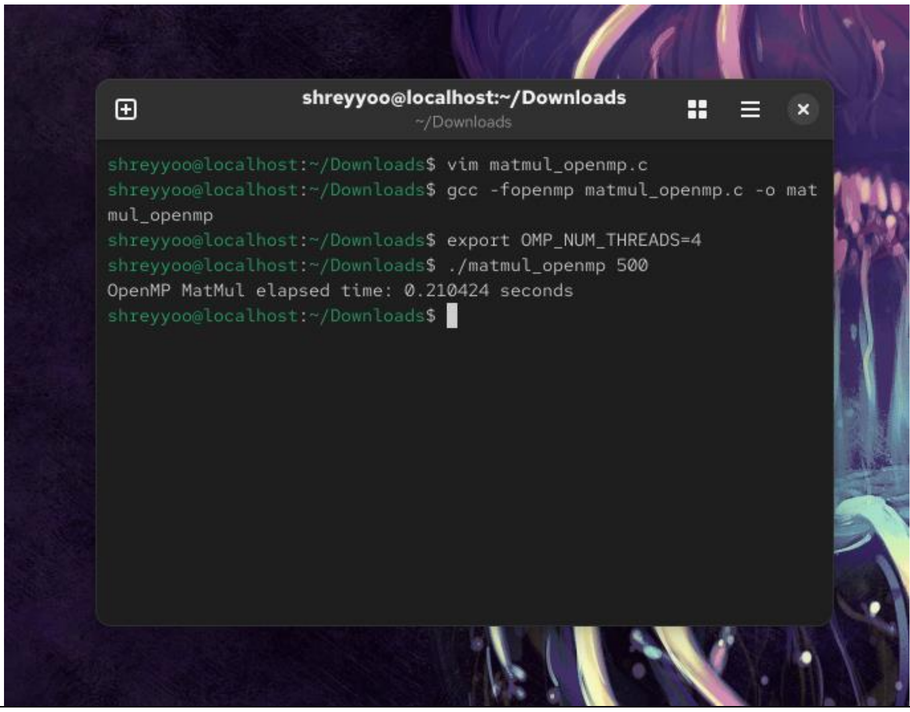
</td></tr>
</table>

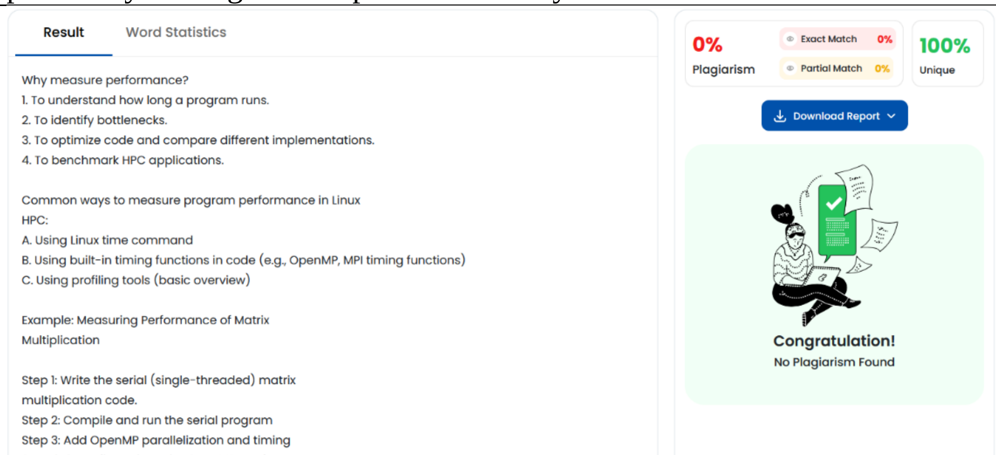
2. Matmul_openmp.c

```
#include <stdio.h>
#include <stdlib.h>
#include <omp.h>

void matmul(int N, double *A, double *B, double *C) {
    #pragma omp parallel for collapse(2)
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++) {
            double sum = 0;
            for (int k = 0; k < N; k++)
                sum += A[i*N+k] * B[k*N+j];
            C[i*N+j] = sum;
        }
}

int main(int argc, char **argv) {
    if (argc < 2) {
        printf("Usage: %s matrix_size\n", argv[0]);
        return 1;
    }
    int N = atoi(argv[1]);
    double *A = malloc(N*N*sizeof(double));
    double *B = malloc(N*N*sizeof(double));
    double *C = malloc(N*N*sizeof(double));

    for (int i = 0; i < N*N; i++) {
        A[i] = 1.0;
        B[i] = 2.0;
    }

    double start = omp_get_wtime();
    matmul(N, A, B, C);
    double end = omp_get_wtime();

    printf("OpenMP MatMul elapsed time: %f seconds\n", end - start);

    free(A); free(B); free(C);
    return 0;
}
```

Output


shreyyoo@localhost:~/Downloads
~/Downloads
shreyyoo@localhost:~/Downloads$ gcc matmul_serial.c -o matmul_serial
shreyyoo@localhost:~/Downloads$ ./matmul_serial 500
Serial MatMul elapsed time: 0.446486 seconds
shreyyoo@localhost:~/Downloads$


shreyyoo@localhost:~/Downloads
~/Downloads
shreyyoo@localhost:~/Downloads$ vim matmul_openmp.c
shreyyoo@localhost:~/Downloads$ gcc -fopenmp matmul_openmp.c -o matmul_openmp
shreyyoo@localhost:~/Downloads$ export OMP_NUM_THREADS=4
shreyyoo@localhost:~/Downloads$ ./matmul_openmp 500
OpenMP MatMul elapsed time: 0.210424 seconds
shreyyoo@localhost:~/Downloads$

Output Analysis	The matmul_serial file executes in 0.4 seconds, whereas the matmul_openmp files executes in 0.2. There's a significate improvement in execution time between the serial matrix multiplication algorithm and the multi-threaded OpenMP matrix multiplication algorithm.
Link of student Github profile where lab assignment has been uploaded	**https://github.com/Sohampimpalgaonkar/HPC**
Conclusion	Using OpenMP we can drastically improve the performance of an algorithm, and we can implement shared-memory parallelism which offers high portability and high development efficiency.
Plag Report (Similarity index < 12%)	
Date	28/10/2025