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 harness the power of artificial intelligence and data science to solve real-world problems and enhance human potential.	**Mission:** To acquire skills through coursework, projects, and internships, while actively engaging in research and collaboration with peers to innovate and apply AI solutions.

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): (statements that describe what a student should be able to do and know by the end of a program)

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.

Prerana Bijekar 28 October 2025
Name and Signature of Student and Date
(Signature and Date in Handwritten)

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	11	**Name of Student**	Prerana Bijekar

Practical Number	4
Course Outcome	**CO1:** Understand and Apply Parallel Programming Concepts **CO2:** Analyze and Improve Program Performance. **CO3:** Demonstrate Practical Skills in HPC Tools and Environments.
Aim	Matrix Multiplication using OpenMP
Theory (100 words)	Measuring program performance involves quantifying a program's efficiency and effectiveness. The main goal is to understand how well a program utilizes resources and if it's achieving its intended goals. Key metrics include execution time (wall-clock time), CPU utilization, and memory usage. The theory is that by systematically collecting and analyzing this data, you can identify bottlenecks, optimize code, and make informed decisions to improve a program's overall performance. This isn't just about making things faster; it's about ensuring your program is a good neighbor in a shared computing environment by not wasting resources.
Procedure and Execution (100 Words)	Algorithm: • Examining the Serial Code: Opened a C source file named • matmul_serial.c using the nano text editor. • Compiling the Serial Code: Compiled the serial code into executable file using command gcc -o matmul_serial matmul_serial.c • Running the Serial Program: Executed the compiled program using command ./matmul_serial 500 • Examining the Parallel Code: Opened a new C source file named • matmul_openmp.c in the nano editor. • Compiling the Parallel Code: Compiled the parallel code using the command gcc -fopenmp -o matmul_openmp matmul_openmp.c • Running the Parallel Program and Analyzing Results:Ran the parallel program using the command ./matmul_openmp 500

Code:

```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;
}
```

```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;
}
```

<table>
<tr><td></td><td>Output:
<pre>[lab1@localhost ~]$ nano matmul_serial.c
[lab1@localhost ~]$ gcc -o matmul_serial matmul_serial.c
[lab1@localhost ~]$ ./matmul_serial 500
Serial MatMul elapsed time: 0.326071 seconds
[lab1@localhost ~]$ nano matmul_openmp.c
[lab1@localhost ~]$ gcc -fopenmp -o matmul_openmp matmul_openmp.c
[lab1@localhost ~]$ ./matmul_openmp 500
OpenMP MatMul elapsed time: 0.033727 seconds
[lab1@localhost ~]$</pre></td></tr>
<tr><td>Output Analysis</td><td>The parallel program was almost 10 times faster than the serial version. This significant speedup, from 0.326 seconds to 0.033 seconds, proves that OpenMP successfully improved performance by distributing the matrix multiplication task across multiple CPU cores.</td></tr>
<tr><td>Github Link</td><td>https://github.com/Prerana-Bijekar/HPC</td></tr>
<tr><td>Conclusion</td><td>This practical proved that parallel computing with OpenMP is highly effective. By comparing the serial and parallel matrix multiplication programs, we saw a massive performance gain, showing that distributing tasks across multiple processor cores is essential for efficient computing.</td></tr>
<tr><td>Plag Report (Similarity index < 12%)</td><td>
</td></tr>
<tr><td>Date</td><td>28 October 2025</td></tr>
</table>