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 Artificial Intelligence & Data Science

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: Dream of where you want.	**Mission:** Means to achieve Vision

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
Sakshi Gokhale
02/09/25

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 Artificial Intelligence & Data Science

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 AIDS	**Course Code**	22ADS706
Roll No	16	**Name of Student**	Sakshi Gokhale

Practical Number	3
Course Outcome	1. Understand and Apply Parallel Programming Concepts 2. Analyse and Improve Program Performance
Aim	Introduction to OpenMP
Problem Definition	Introduction to OpenMP
Theory (100 words)	OpenMP (Open Multi-Processing) is an Application Programming Interface (API) that provides a simple and flexible way to develop parallel applications for shared-memory architectures. It is widely supported in programming languages such as C, C++ and Fortran, and is particularly useful in high-performance computing environments. OpenMP uses a shared-memory model, where all threads created during program execution have access to the same memory space. This allows for efficient data sharing among threads without the need for explicit message passing. The programming model is based on the fork-join principle, where the main (master) thread forks multiple worker (slave) threads to perform parallel tasks, and once the work is done, they rejoin back into a single thread. One of the main strengths of OpenMP is its simplicity. It uses compiler directives (pragma statements), runtime library routines, and environment variables to control the execution of parallel code. For example, the environment variable OMP_NUM_THREADS allows users to control how many threads are used for execution. OpenMP is highly scalable on multi-core processors, making it a preferred

	choice for scientific computations, simulations, and applications that require intensive numerical calculations. Its directive-based approach allows developers to parallelize existing serial code with minimal modifications, making it easier to learn and apply in practical scenarios.

Procedure and Execution
 	
(100 Words)
	Code:
	
	#include <stdio.h>
	#include <omp.h>
	
	int main() {
	  int i;
	  int n = 10;
	  int a[10];
	
	  // Parallelize loop using OpenMP
	  #pragma omp parallel for
	  for (i = 0; i < n; i++) {
	  a[i] = i * i; // Compute square of index
	  printf("Thread %d: a[%d] = %d\n", omp_get_thread_num(), i, a[i]);
	  }
	
	  return 0;
	}
	
	OpenMP Scheduling Clause
	
	#include <stdio.h>
	#include <omp.h>
	
	int main() {
	  int i;
	  int n = 12;
	
	  // Parallel loop with static scheduling (chunk size = 3)
	  #pragma omp parallel for schedule(static, 3)
	  for (i = 0; i < n; i++) {
	  printf("Thread %d processing iteration %d\n", omp_get_thread_num(), i);
	  }
	
	  return 0;
	}

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.

Using Barrier for Synchronization

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

int main() {
   // Create 4 threads
   #pragma omp parallel num_threads(4)
   {
      int id = omp_get_thread_num();

      printf("Thread %d before barrier\n", id);

      // Barrier: all threads must reach here before proceeding
      #pragma omp barrier

      if (id == 0) {
         printf("All threads reached the barrier. Thread %d continuing.\n", id);
      }
   }

   return 0;
}
```

Nested Parallelism

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

int main() {
   // Enable nested parallel regions
   omp_set_nested(1);

   // Outer parallel region with 2 threads
   #pragma omp parallel num_threads(2)
   {
      int id = omp_get_thread_num();
      printf("Outer thread %d starting\n", id);

      // Inner parallel region with 2 threads per outer thread
      #pragma omp parallel num_threads(2)
      {
```

Vision of the Department

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

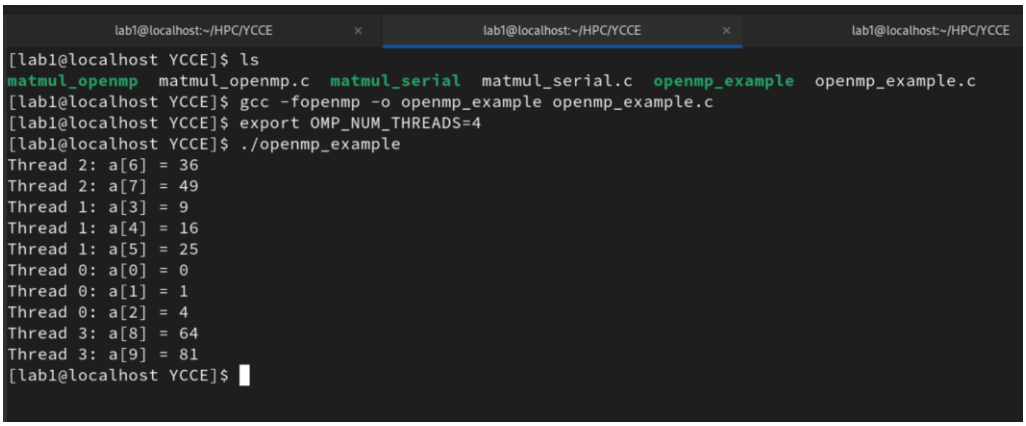
Mission of the Department

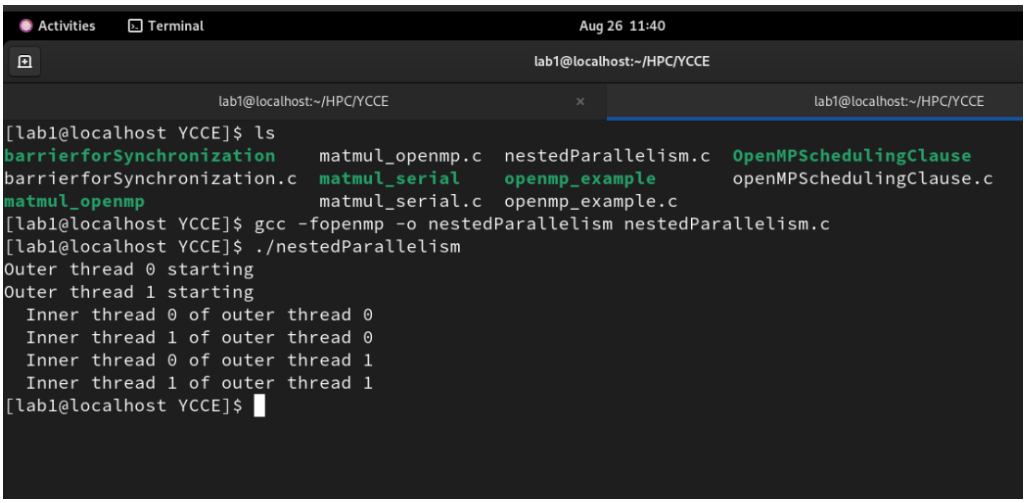
To establish learning ambience for ushering in computer engineering professionals in core and multidisciplinary area by developing Problem-solving skills through emerging technologies.

```
            int inner_id = omp_get_thread_num();
            printf("  Inner thread %d of outer thread %d\n", inner_id, id);
        }
    }

    return 0;
}
```

Output:



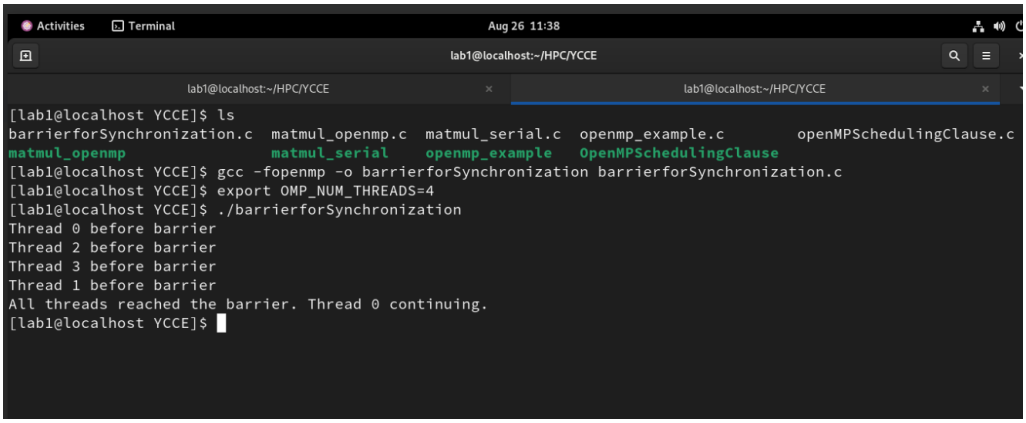


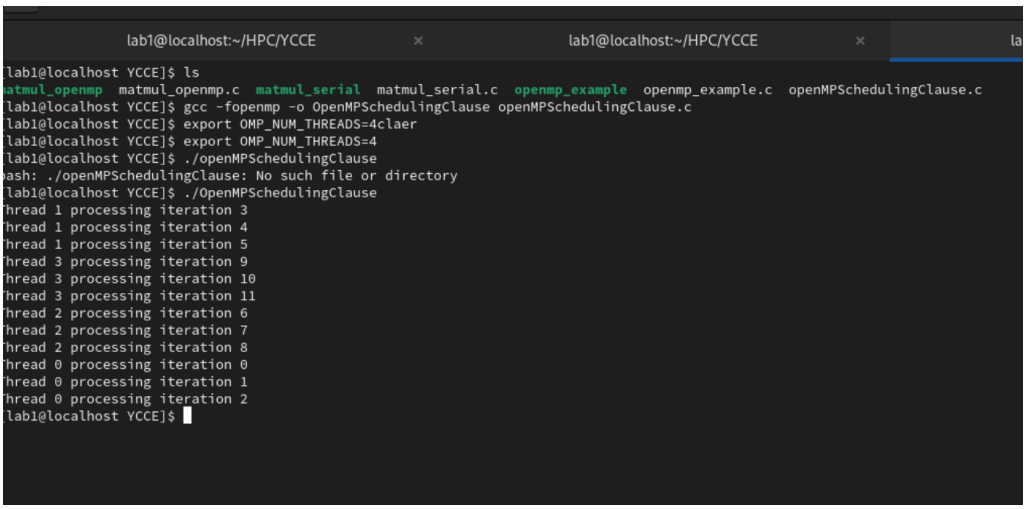
Vision of the Department

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.

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




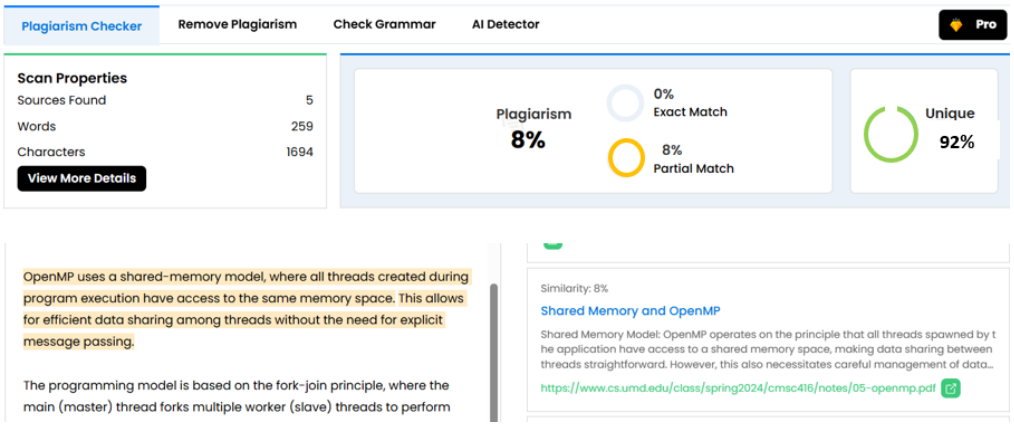
</td></tr>
<tr><td>Output Analysis</td><td>When the program is executed, the array elements are computed in parallel using multiple threads. Each thread is assigned a subset of loop iterations, and it calculates the square of the index value. The output displays messages such as Thread 1: a[3] = 9 or Thread 0: a[0] = 0, showing which thread processed which iteration. Since the loop is executed in parallel, the order of the printed lines may not be sequential, but the final values stored in the array remain correct.
This confirms that OpenMP successfully distributes the work among available threads to perform the computation efficiently.</td></tr>
<tr><td>Link of student</td><td>https://github.com/sakshi-gokhale/Lab-HPC</td></tr>
</table>

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.

Github profile where lab assignment has been uploaded	
Conclusion	The experiment demonstrates how OpenMP can be used to parallelize simple loops with minimal code changes. By applying the #pragma omp parallel for directive, the workload is divided among multiple threads, reducing execution time for larger problems. Although the output order is non-deterministic due to concurrent execution, the correctness of the results is maintained. This practical highlights the ease of implementing parallelism with OpenMP and its effectiveness in utilizing multi-core processors for computational tasks.
Plag Report (Similarity index < 12%)	
Date	02/09/25