**Class:** Third Year B. Tech (Computer Science and Engineering)

**Year:** 2025-26 **Semester:** Odd

**Course:** Cutting Edge Technologies Lab

**Course Code:** 7CS352

**Practical No. 2**

PRN: 23510030

Name: Harshvardhan Ganapati Mane

Batch:T1

**Title of practical: Parallel region creation, thread identification using OpenMP.**

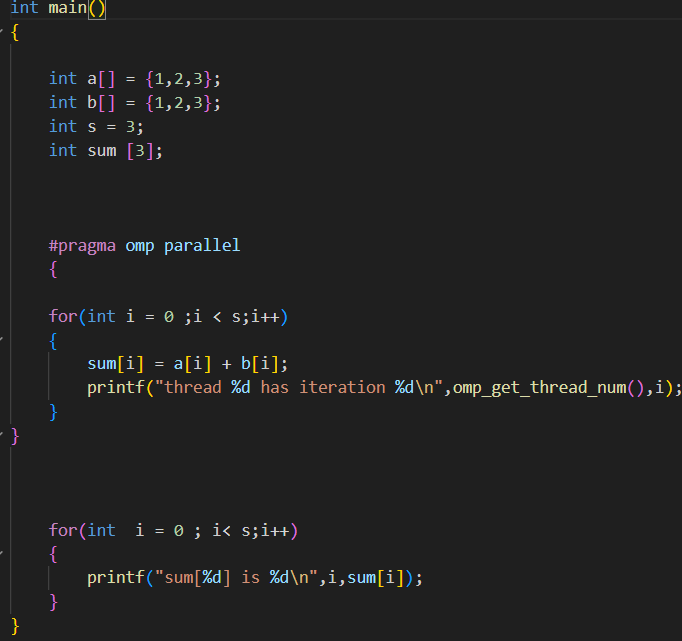
Implement following Programs using OpenMP with C:

1. Vector Vector Addition

Analyse the performance of your programs for different number of threads and Data size. Understand different ways in which you can set the number of threads for execution of parallel section, how to access thread ID and total number of threads in a code.

**Problem Statement 1:**

**Screenshots:**

**  
Code:**

**#include<stdio.h>**

**#include<omp.h>**

**int main()**

**{**

**int a[] = {1,2,3};**

**int b[] = {1,2,3};**

**int s = 3;**

**int sum [3];**

**#pragma omp parallel**

**{**

**for(int i = 0 ;i < s;i++)**

**{**

**sum[i] = a[i] + b[i];**

**printf("thread %d has iteration %d\n",omp\_get\_thread\_num(),i);**

**}**

**}**

**for(int  i = 0 ; i< s;i++)**

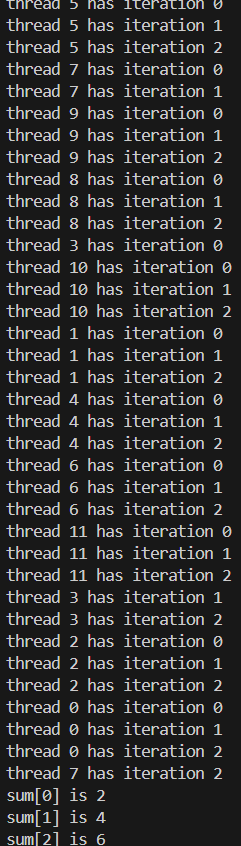
**{**

**printf("sum[%d] is %d\n",i,sum[i]);**

**}**

**}**

**Output:**



**Information:**

The program demonstrates two approaches for vector addition. The first, using #pragma omp parallel for, divides the five addition operations among four threads, resulting in five total print statements. The second, using only #pragma omp parallel, causes each of the four threads to perform all five additions, resulting in twenty total print statements.

**Analysis:**

**Significant Speedup**: For computationally heavy tasks like matrix multiplication, parallelization provides a major performance boost, running over **3.5 times faster** than the serial version. This is the primary benefit of OpenMP.

**Github Link:**