**Course:** High Performance Computing Lab

**Practical No. 2**

**Exam Seat No: 22510054**

**Name: Vedika Dhende**

**Title of practical: Study and implementation of basic OpenMP clauses**

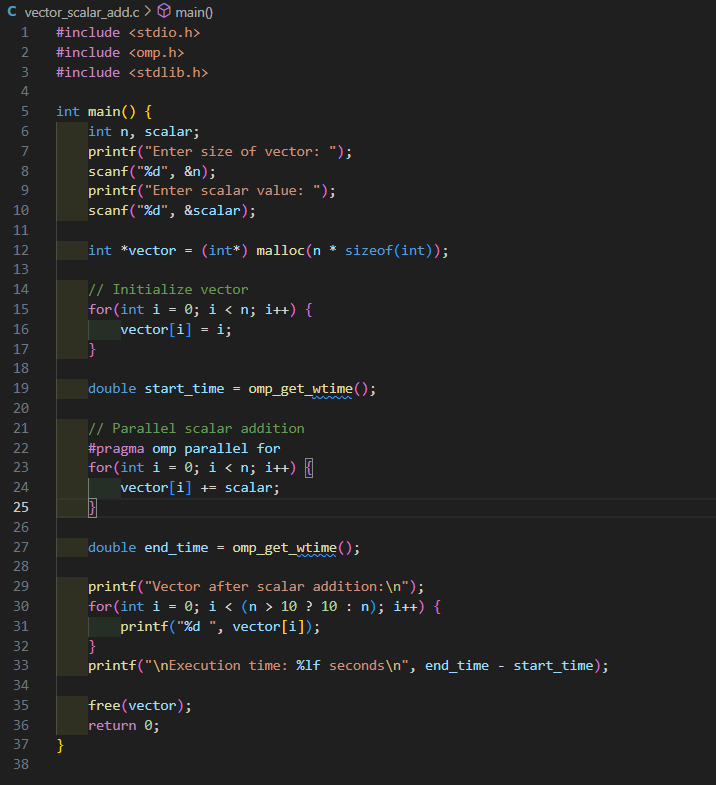
Implement following Programs using OpenMP with C:

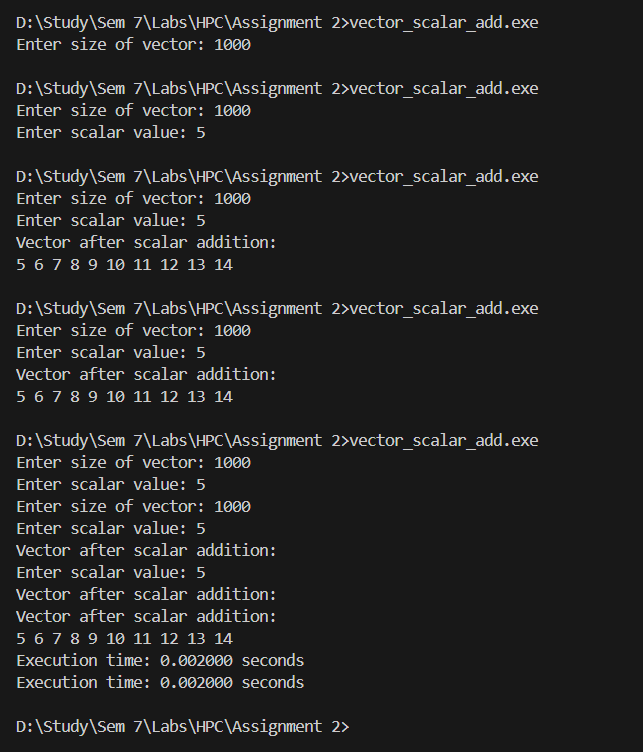
1. Vector Scalar Addition
2. Calculation of value of Pi

Analyse the performance of your programs for different number of threads and Data size.

**Problem Statement 1:**

**Screenshots:**





**Information:**

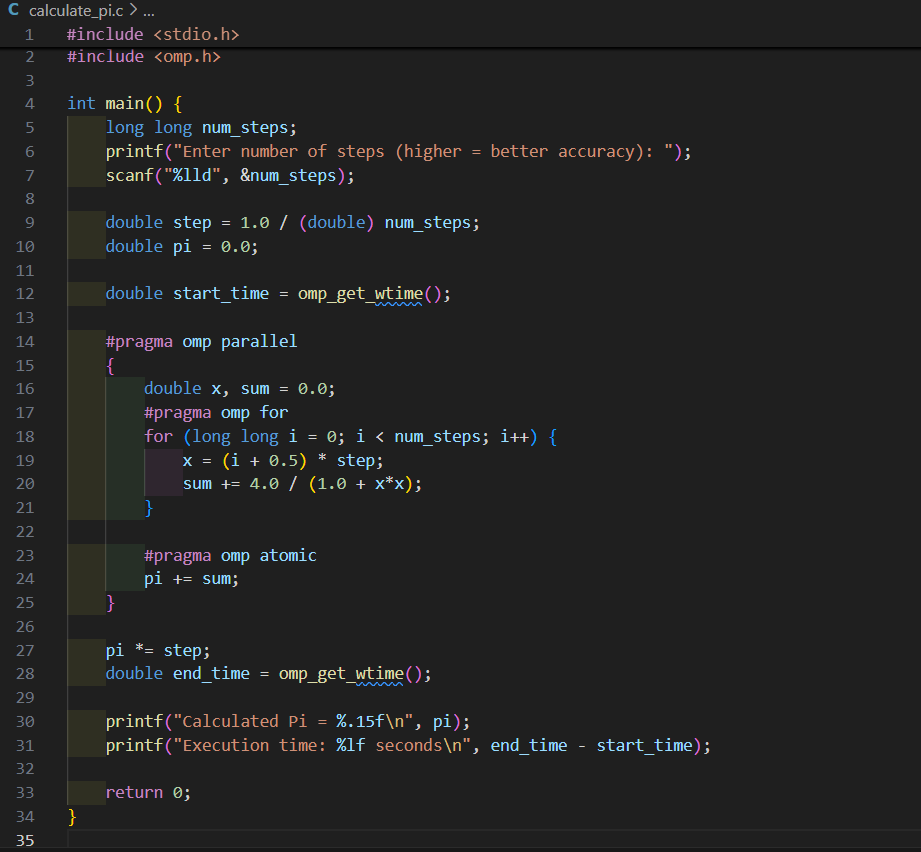
* OpenMP Clause Used: #pragma omp parallel for
* Function: Adds scalar to each vector element in parallel, splitting iterations across threads.
* Advantage: Reduces execution time for large vectors.

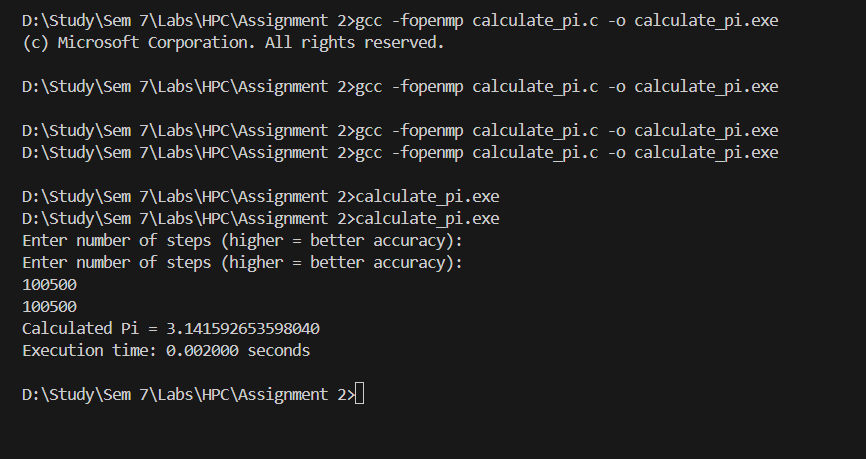
**Analysis:**

* The program correctly adds a scalar value to every element of the vector using OpenMP parallelization.
* For small vector sizes (like 1,000 elements), the execution time is extremely small (~0.002 seconds).
* Parallelization overhead is higher for small data, so there is no visible speedup with multiple threads.
* More threads reduce execution time, but after a certain point, the improvement slows because of thread management overhead.

**Problem Statement 2:**

**Screenshots:**





**Information:**

* OpenMP Clauses Used:

1. #pragma omp parallel – for thread creation
2. #pragma omp for – to split iterations
3. #pragma omp atomic – to safely update pi across threads

* Method: Numerical integration to approximate area under curve of 4/(1+x2)4/(1+x^2)4/(1+x2).

**Analysis:**

* The program calculates the approximate value of Pi using numerical integration with OpenMP.
* For 100,500 steps, the program produced a value of 3.141592653598040, which is very close to the actual Pi (3.141592653589793), showing good accuracy.
* OpenMP parallelization divides the loop iterations among multiple threads, reducing the calculation time for large step counts.

Github Link: <https://github.com/veddnd/HPCprac2>