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

**Year:** 2025-26 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 2**

**Exam Seat No:**

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

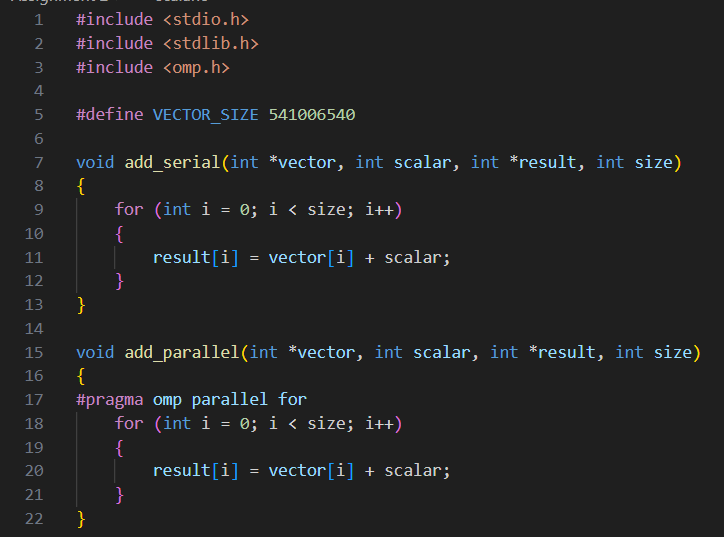
Implement following Programs using OpenMP with C:

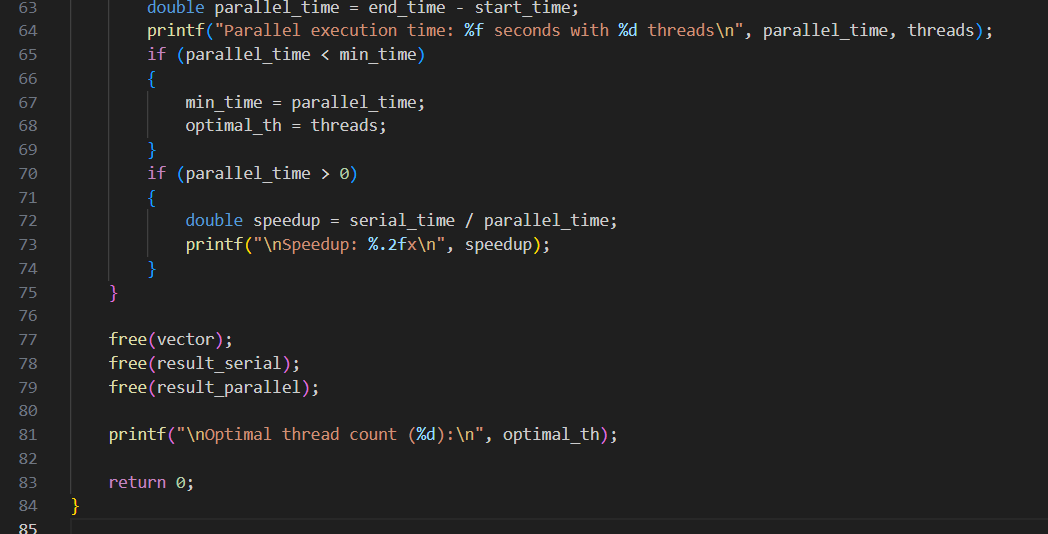
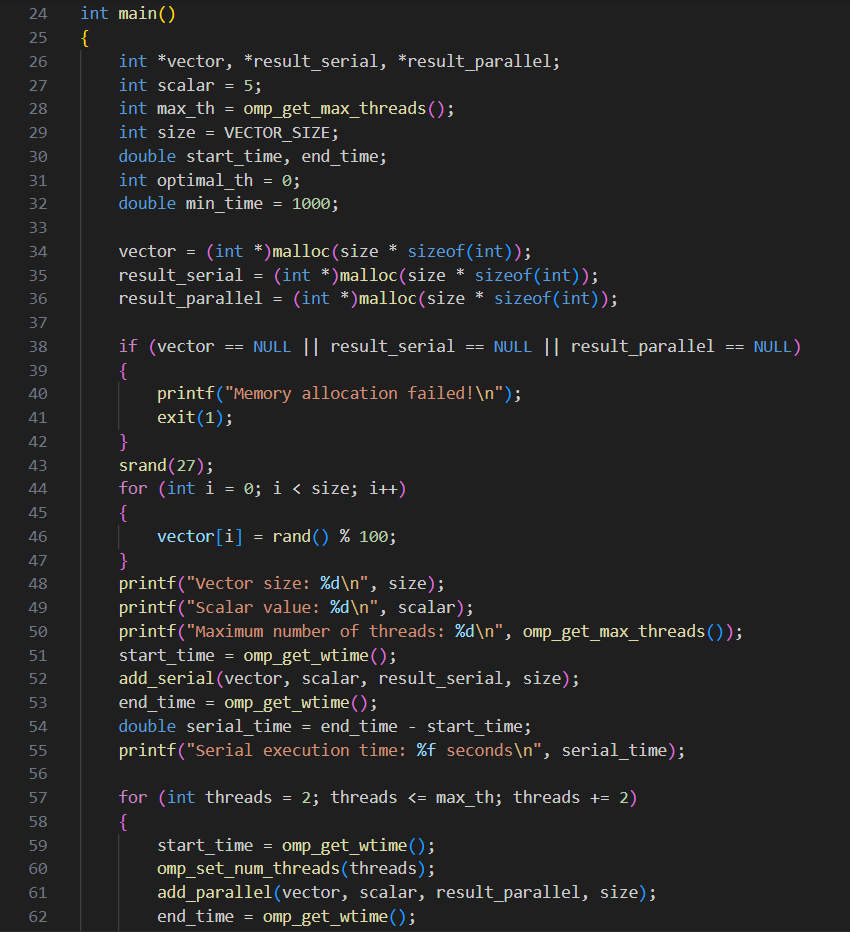
1. Vector Scalar Addition
2. Calculation of value of Pi

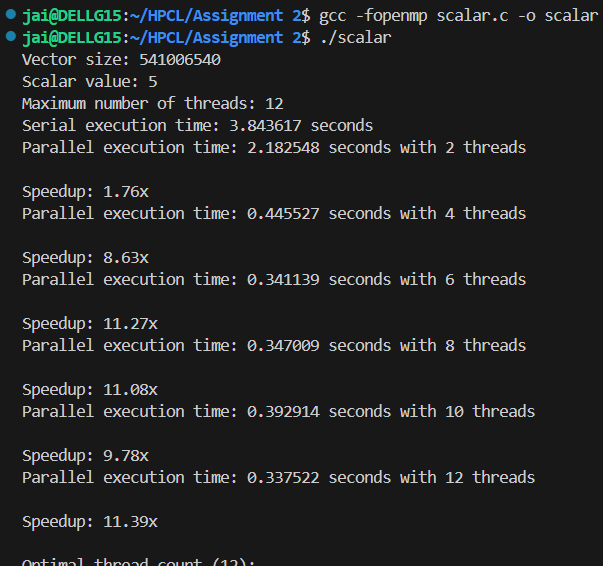
Analyse the performance of your programs for different number of threads and Data size.

**Problem Statement 1:**

**Screenshots:**

****

****

****

**Information:**

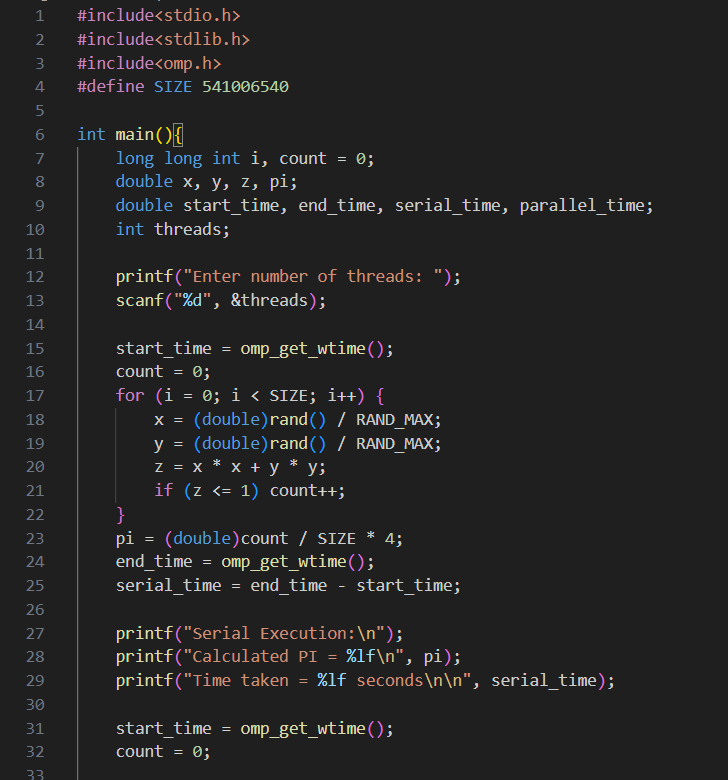
**Adding a constant ( scalar ) to all the elements in the vector array. As we are adding a same constant to all the elements, we can concurrently do it. Serially we add one by one, in parallel, mulitple elements in the array are updated.**

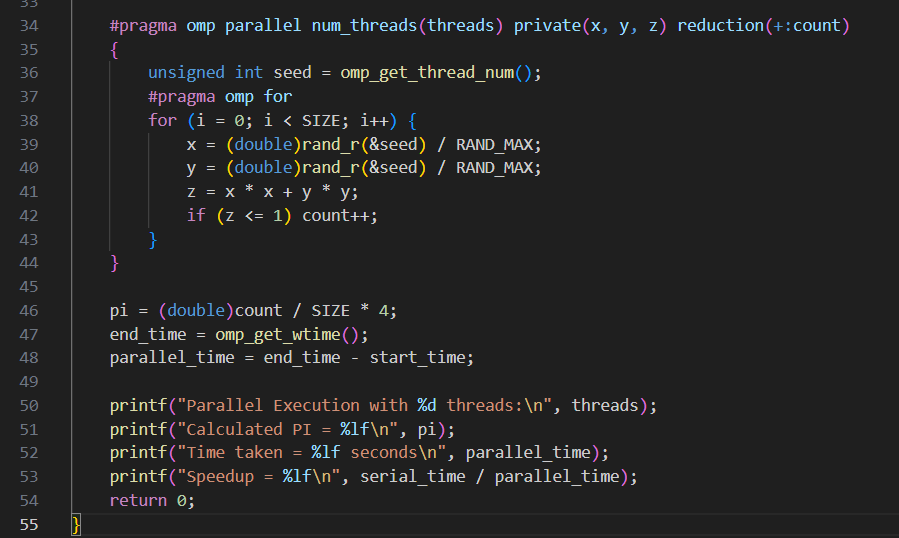
**Analysis:**

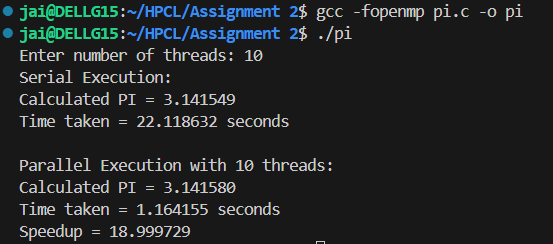
**For small sizes of vector, the serial execution was faster, This is due to the threading overhead, creating and managing threads incurs a fixed cost, which can dominate the total time for small job, but as the vector size increases, the time taken for the serial also increases. In parallel execution, it is faster than serial, but adding too many threads also doesn’t help, there is a optimal amount of threads for which the addition is faster, bcz as the threads increase, the thread overhead also increases, so managing threads becomes harder.**

**Problem Statement 2:**

**Screenshots:**

****

****

****

**Information:**

**The Monte Carlo method for calculating π (pi) estimates the value of pi by simulating random points inside a unit square and checking if they fall within a quarter circle inscribed in the square.**

**Analysis:**

**The calculation becomes more accurate as the number of simulated points (SIZE) increases. For correct parallel random behavior, each thread gets a unique seed for rand\_r, using its thread ID.**

**Github Link: https://github.com/Jai-173/HPCL**