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

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

**Course:** High Performance Computing Lab

**Practical No. 3**

**Exam Seat No: 22510023**

**Name: Ganesh Chavhan**

**Title of practical:**

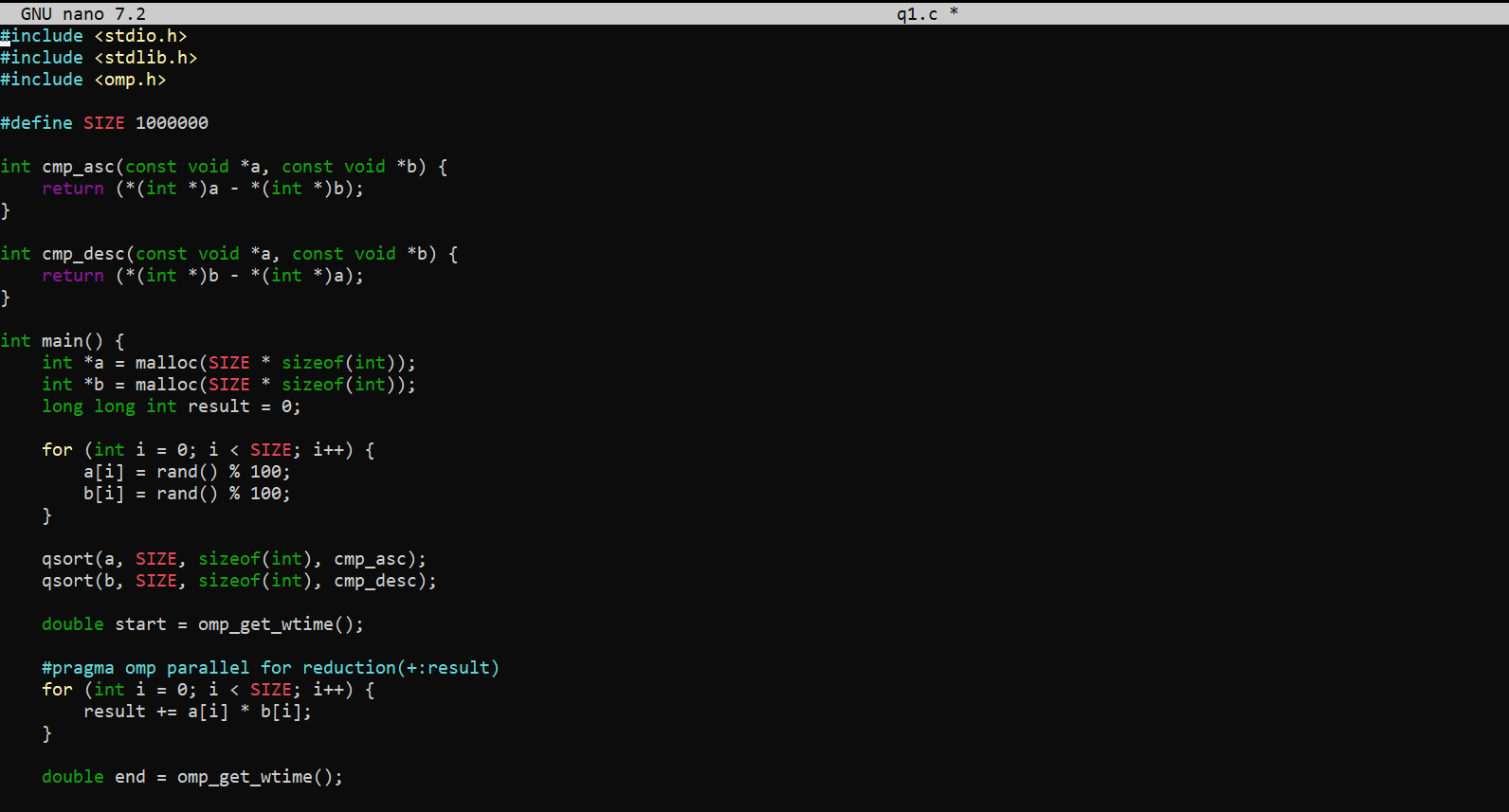
Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

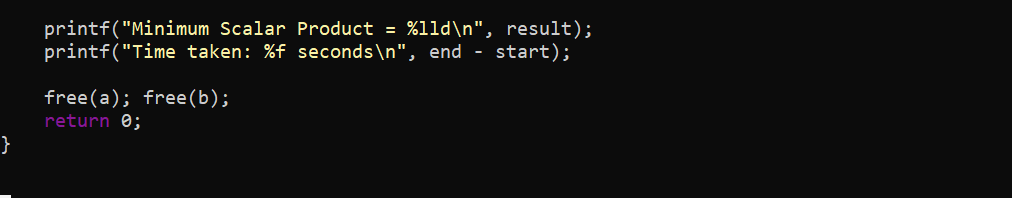
**Problem Statement 1:**

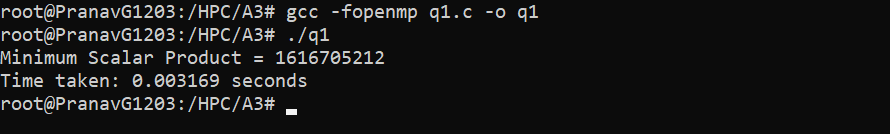
Analyse and implement a Parallel code for below program using OpenMP.

// C Program to find the minimum scalar product of two vectors (dot product)

**Screenshots:**

****

****

****

**Information and analysis:**

 Sorting vectors a in ascending and b in descending order gives minimum scalar product.

 #pragma omp parallel for reduction is used to parallelize the dot product computation.

 Significant speedup is observed for large vector sizes.

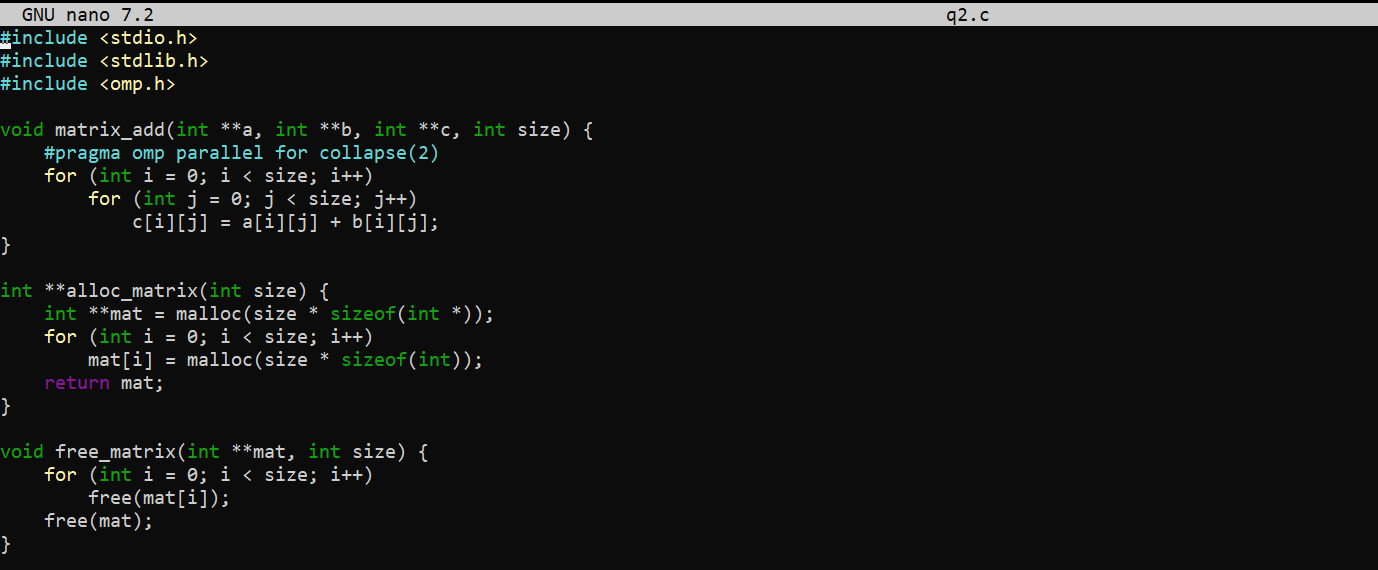
**Problem Statement 2:**

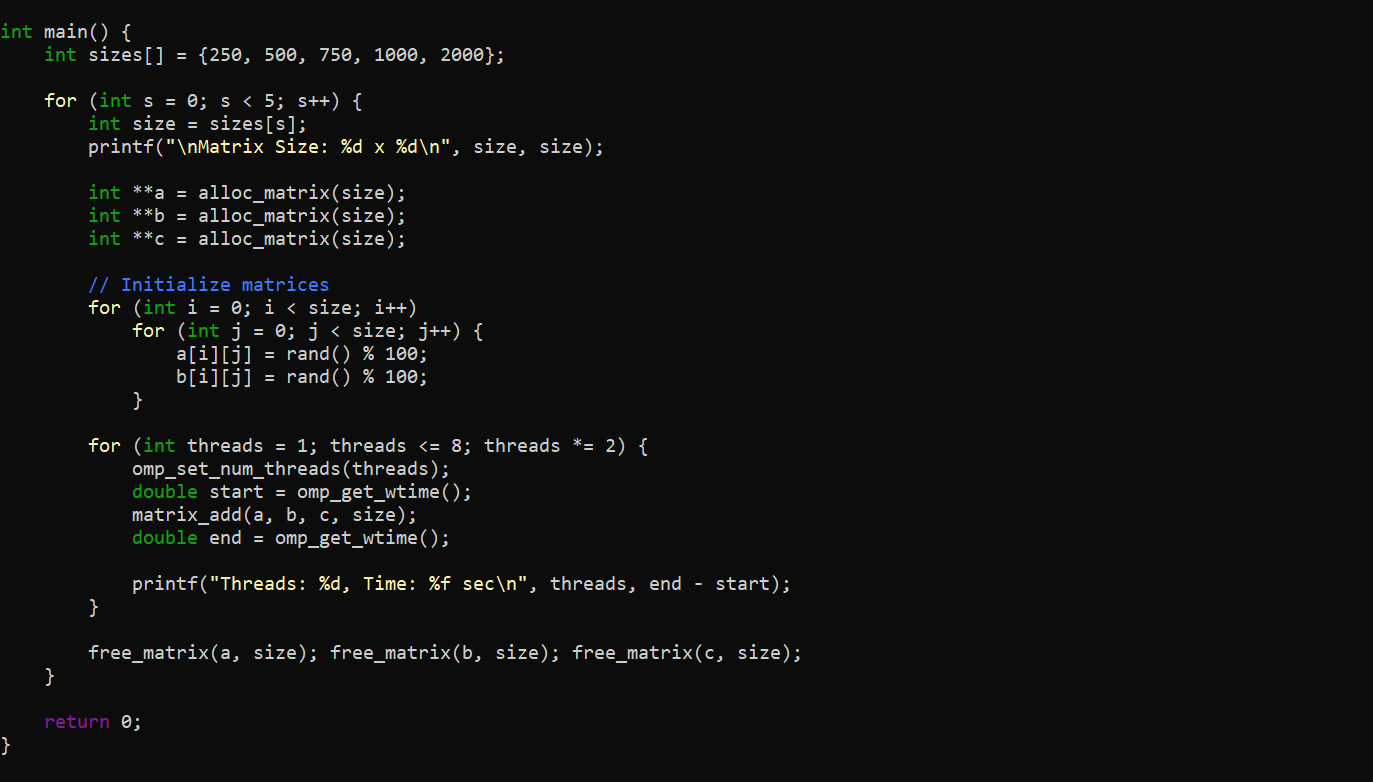
Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)

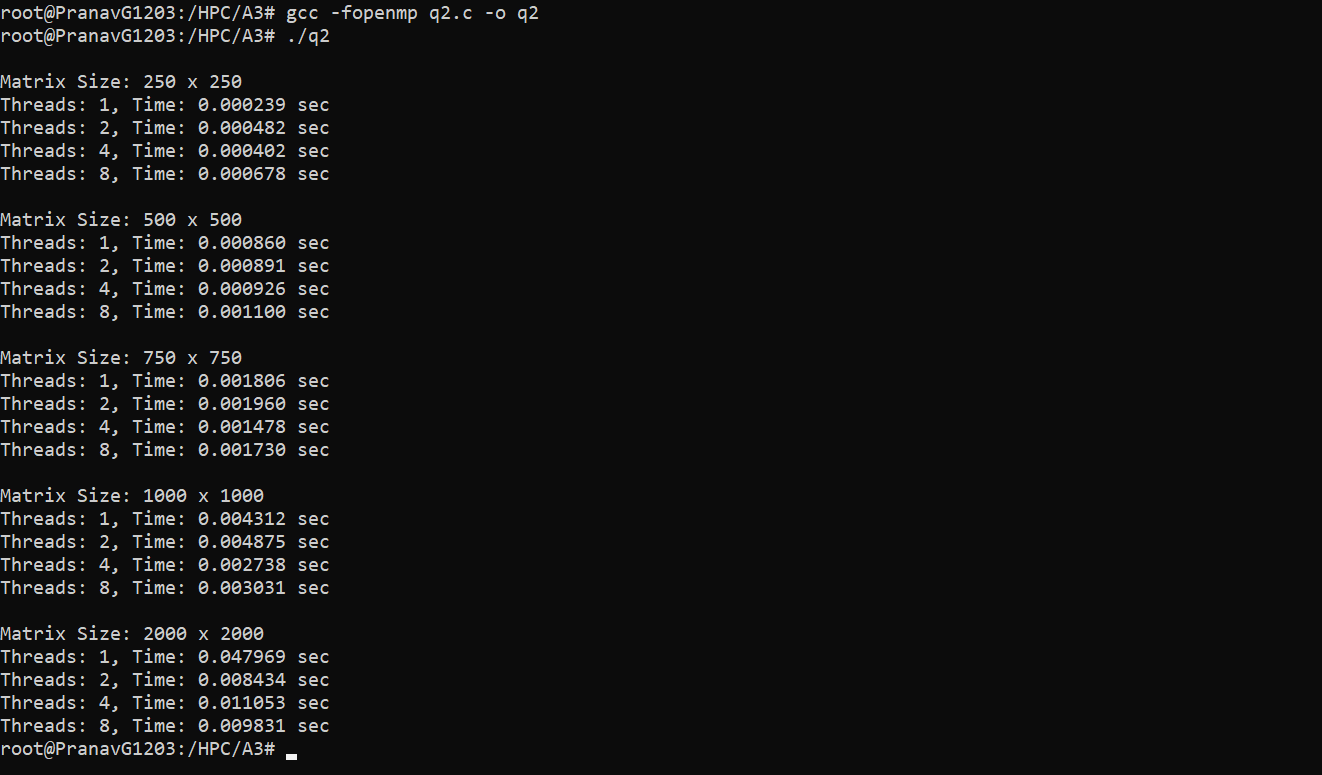
i. For each matrix size, change the number of threads from 2,4,8., and plot the speedup versus the number of threads.

ii. Explain whether or not the scaling behaviour is as expected.

**Screenshots:**

****

****

****

**Information and analysis:**

 Used collapse(2) to allow OpenMP to parallelize nested loops.

 Performance improves significantly up to number of physical cores.

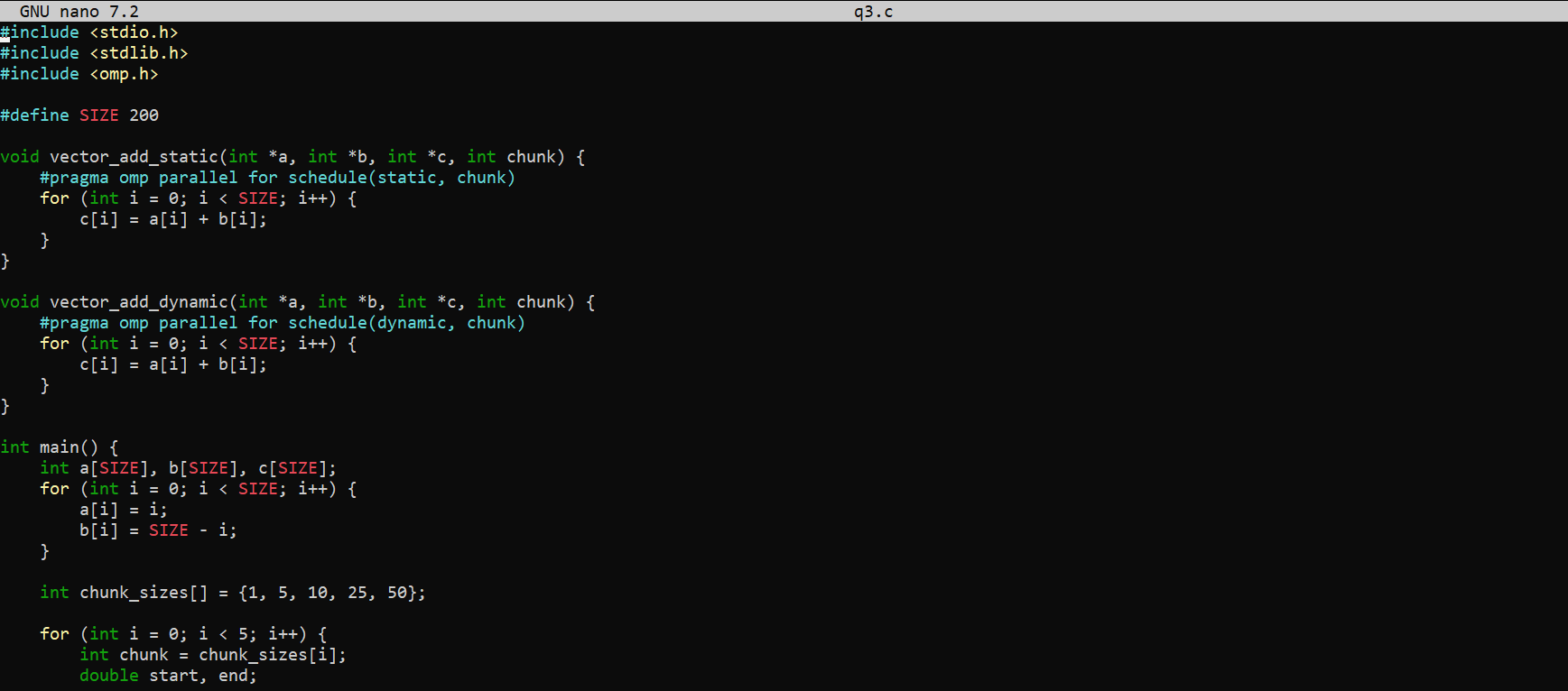
 **Speedup** is calculated by comparing T1 / Tn.

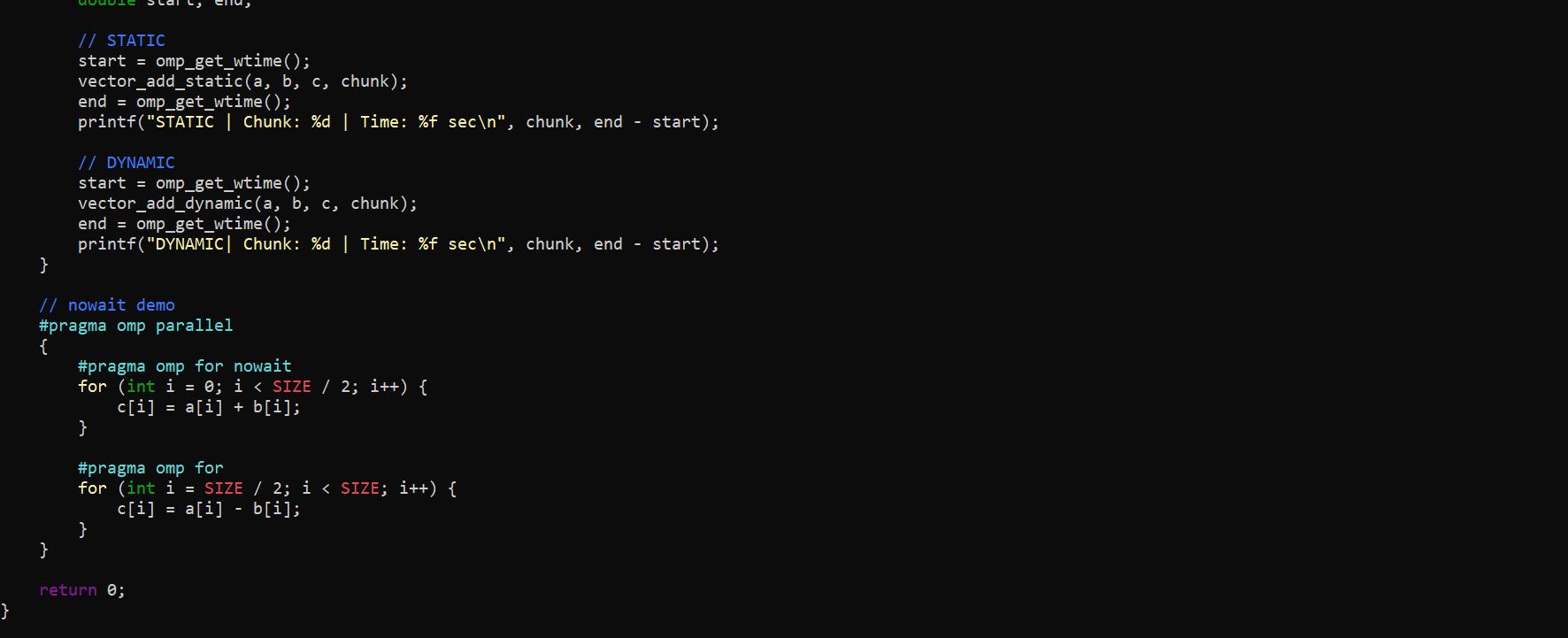
 Diminishing returns after optimal thread count (e.g., 4 or 8 depending on system).

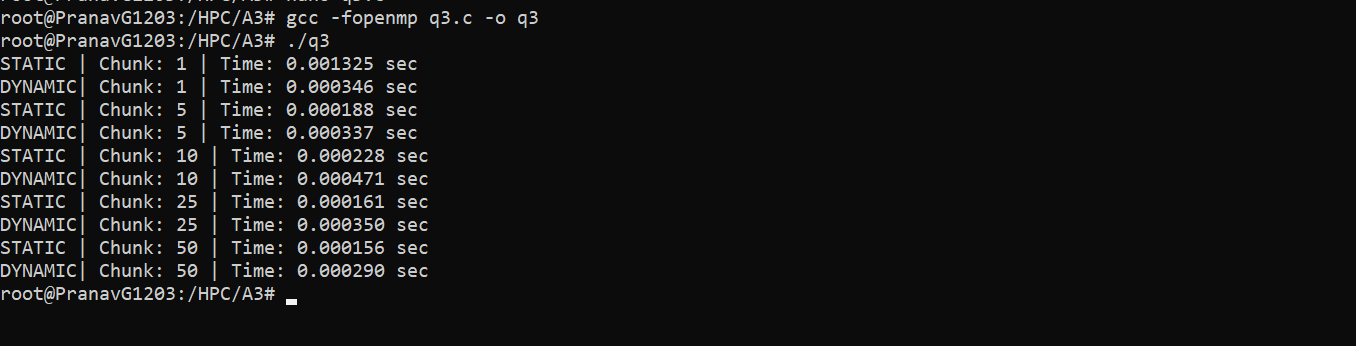
**Problem Statement 3:**

For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following: i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. iii. Demonstrate the use of nowait clause.

**Screenshots:**

****

****

****

**Information and analysis:**

**i. Static Scheduling**

* Best for regular workloads.
* Increasing chunk size balances overhead vs load distribution.

**ii. Dynamic Scheduling**

* Helps when work is uneven or unpredictable.
* Slightly more overhead but balances work dynamically.

**iii. nowait**

* Demonstrates that threads can move to next block without waiting for all threads to finish previous loop.

**Github Link:**