Final Year B.Tech. (CSE) – VII [2024-25]

**6CS452: High Performance Computing Lab**

Assignment No: 3

# Date: 12/08/2024

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

**PRN:** 21510017  **Name:** Onkar Anand Yemul

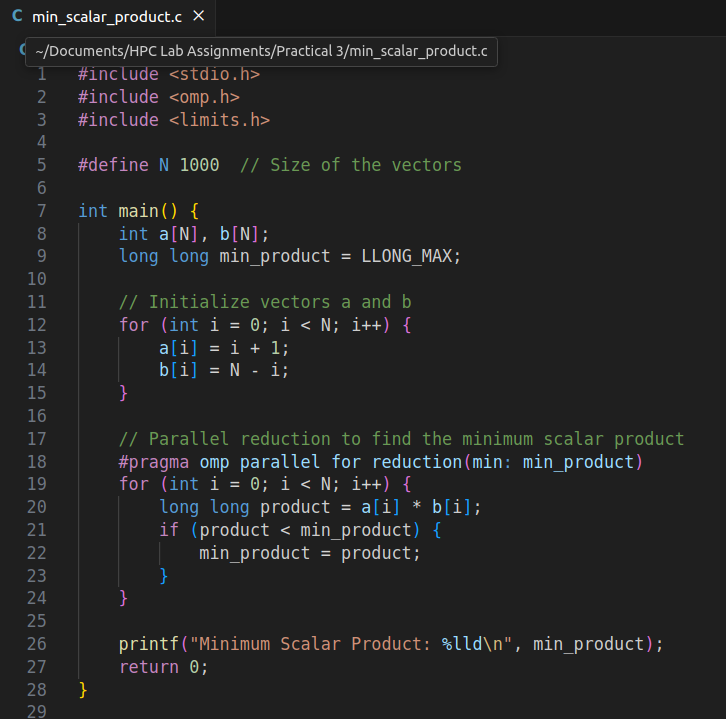
**Title:** Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

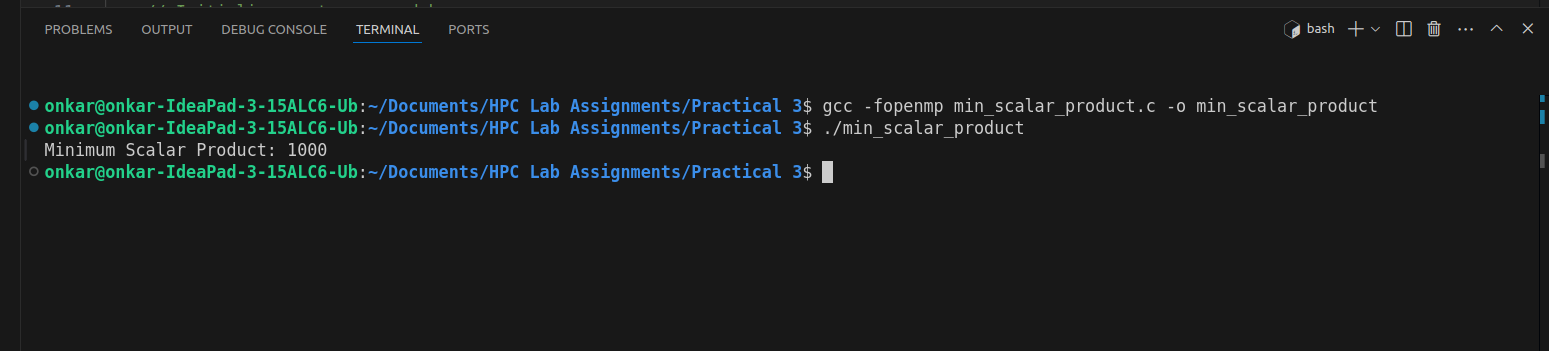
**Problem Statement 1:** Parallel Code for Minimum Scalar Product of Two Vectors

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:**

* Reduction Clause: The reduction clause is used to minimize the product values across all threads, ensuring correct parallel computation.
* Performance: Evaluate the program’s performance by varying the number of threads and comparing execution times.

**------------------------------------------------------------------------------------------------**

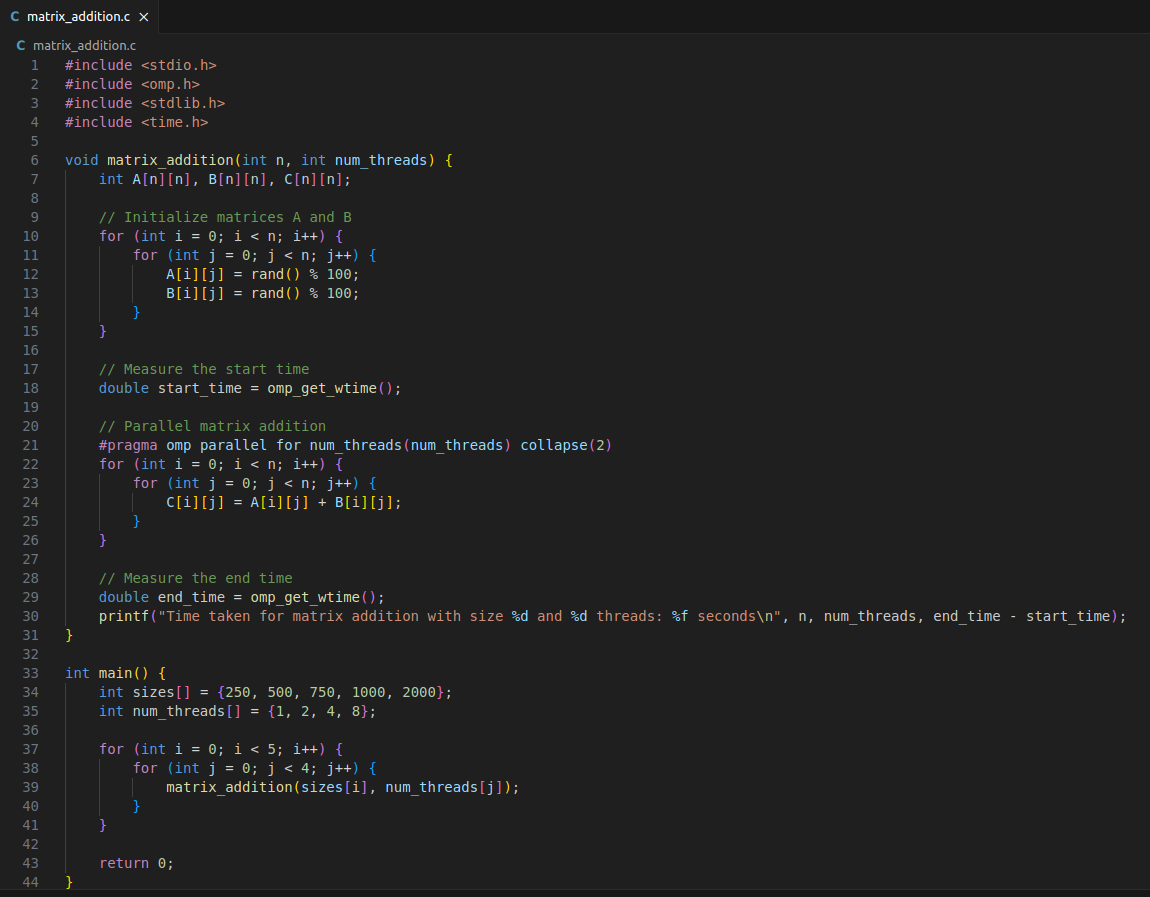
**Problem Statement 2: 2D Matrix Addition with Varying Matrix Sizes**

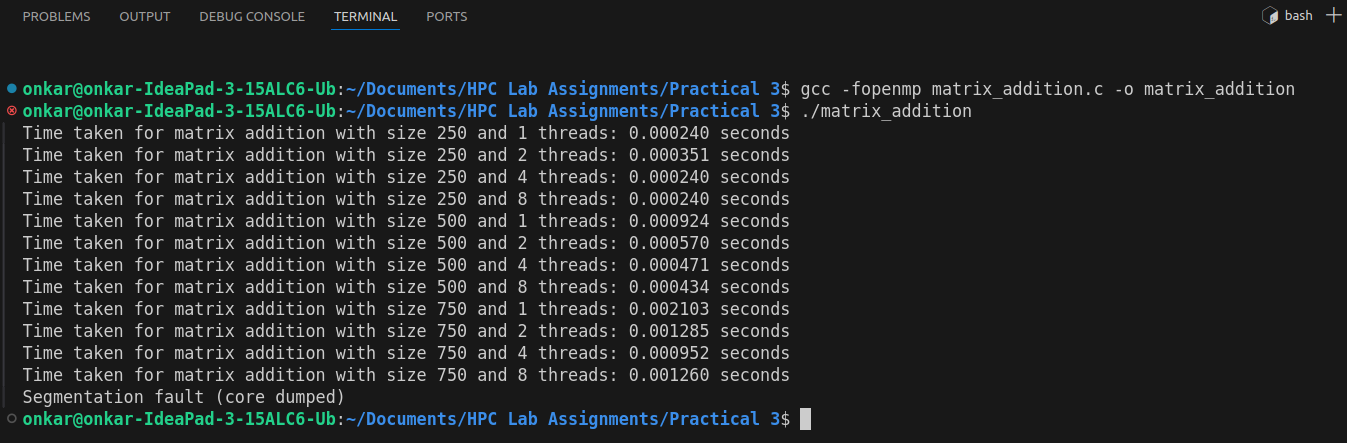
Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)

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:**

* Speedup Plot: Plot the speedup versus the number of threads for each matrix size.
* Scaling Behavior: Analyze if the speedup is linear or deviates due to overhead, memory bandwidth, or thread contention.

**------------------------------------------------------------------------------------------------**

**Problem Statement 3:** 1D Vector and Scalar Addition with STATIC and DYNAMIC Scheduling

For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following:

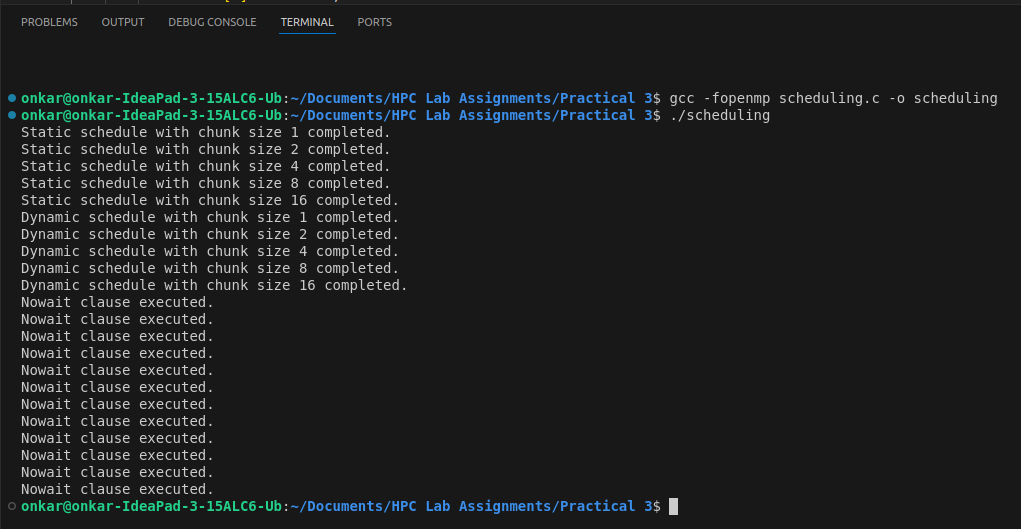
i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup.

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:**

* Static vs. Dynamic: Compare the performance with static and dynamic scheduling for different chunk sizes.
* Nowait Clause: Demonstrate the effect of the nowait clause by analyzing execution times without waiting at the barrier.

**Github Link:**