Course: High Performance Computing Lab

Practical No 3

PRN: 22510054

Name: Vedika Dhende

Batch: B3

**Title of practical:**

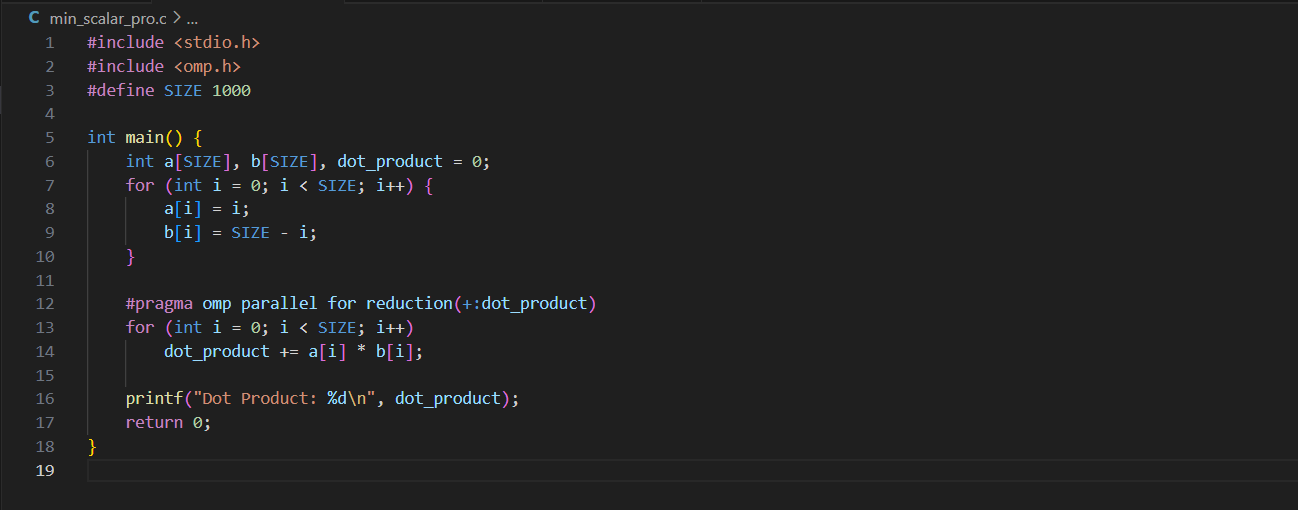
Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

**Problem Statement 1:**

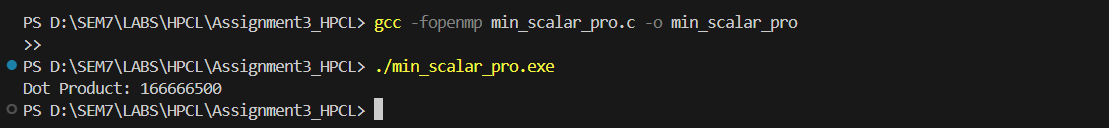
Analyse and implement a Parallel code for below program using OpenMP.

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

**Screenshots:**

****

**Output:**

****

**Information and analysis:**

**OpenMP Concepts Used:**

1. reduction:
   1. Used to safely add up the results from all threads without errors.
2. parallel for:
   1. Used to run the loop in parallel using multiple threads.

**Analysis:**

1. The program runs faster for large input sizes because it uses multiple threads with OpenMP.
2. The reduction makes sure that all thread results are added correctly without conflict.
3. For small inputs, there is no big speedup because starting threads takes some time, which cancels out the benefits.

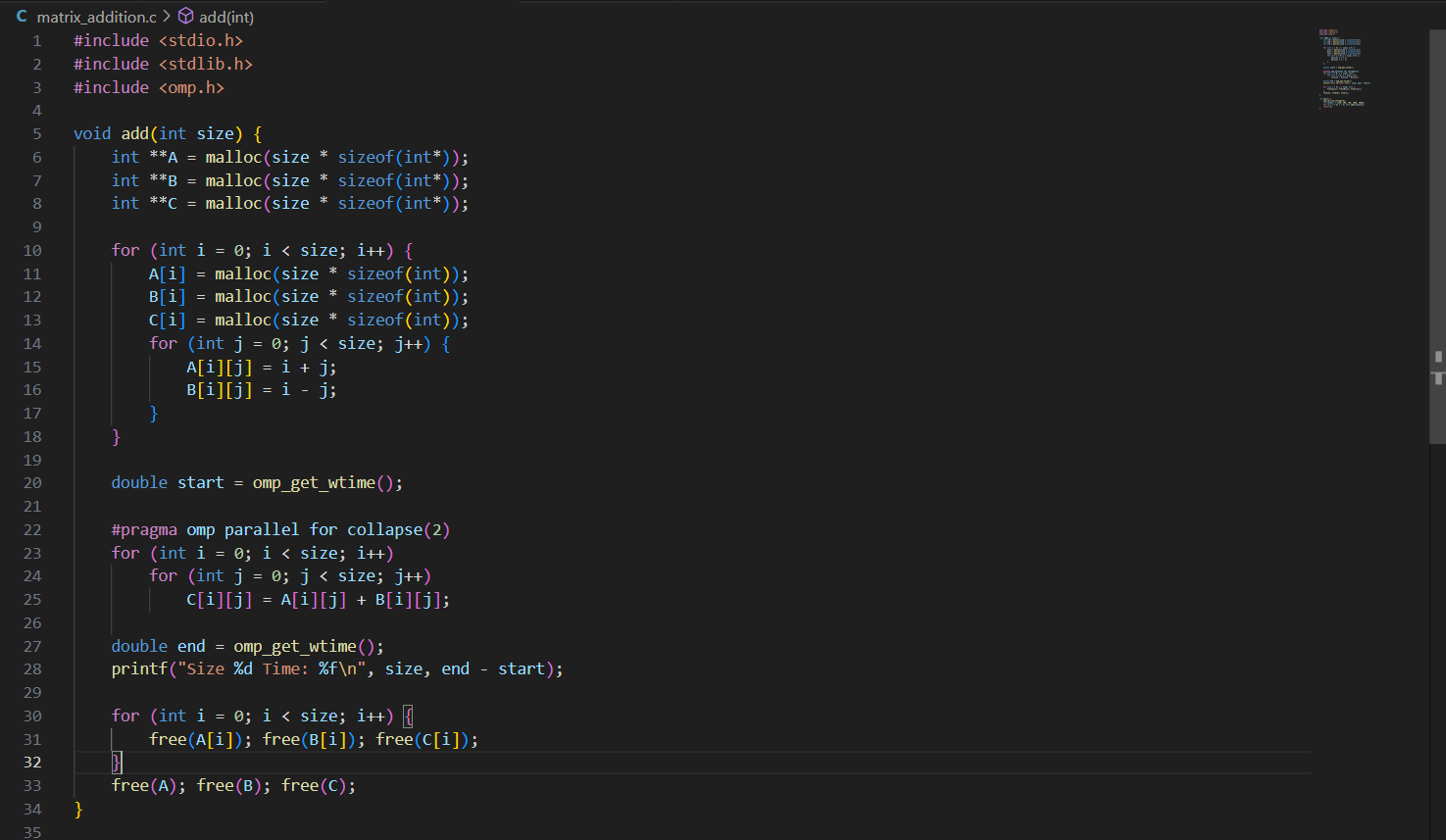
**Problem Statement 2:**

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.

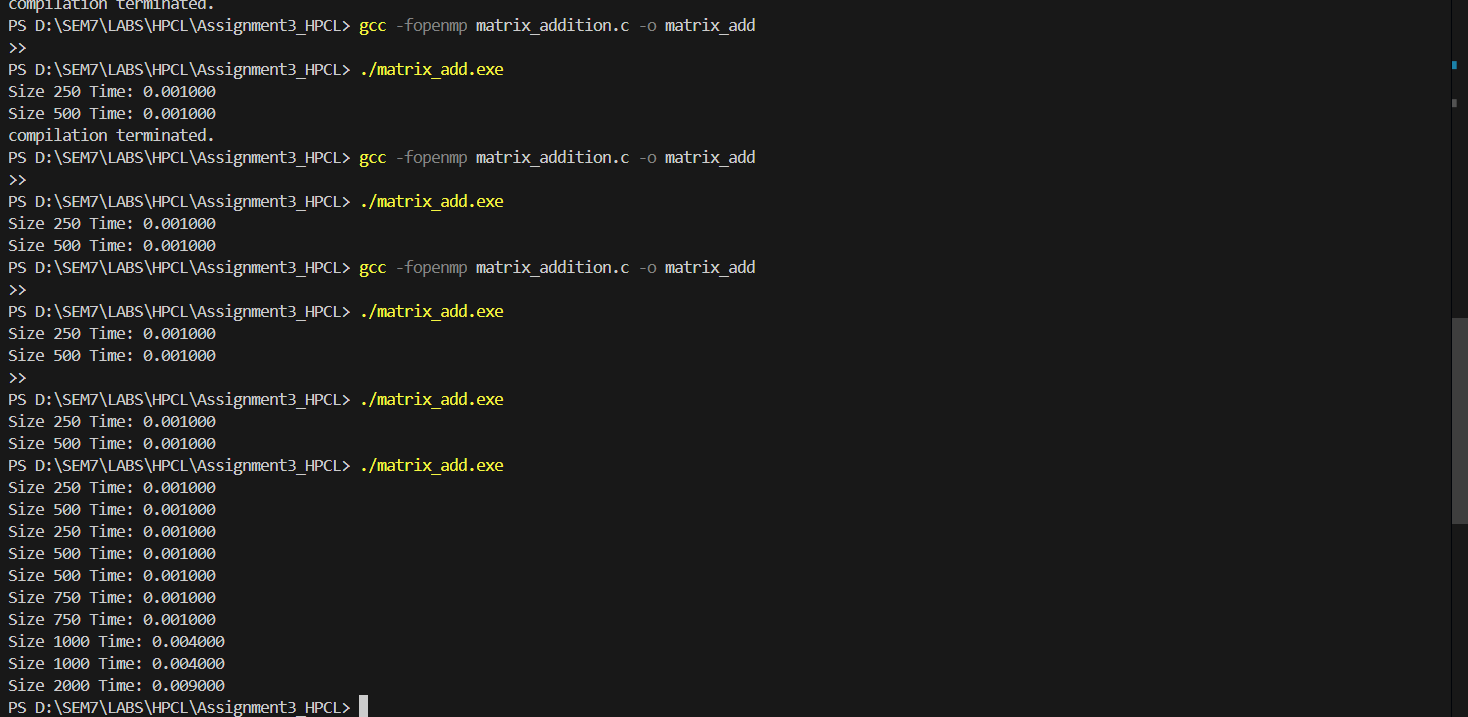
ii. Explain whether or not the scaling behaviour is as expected.

**Screenshots:**

****

****

**Output:**

****

**Information and analysis:**

* **OpenMP Concepts Used:**

1. parallel for:

Used to divide the matrix addition work among threads.

1. collapse(2):

Combined two nested loops to give OpenMP more chunks to distribute evenly.

1. schedule:

Tried different ways to assign loop work to threads to see which is faster.

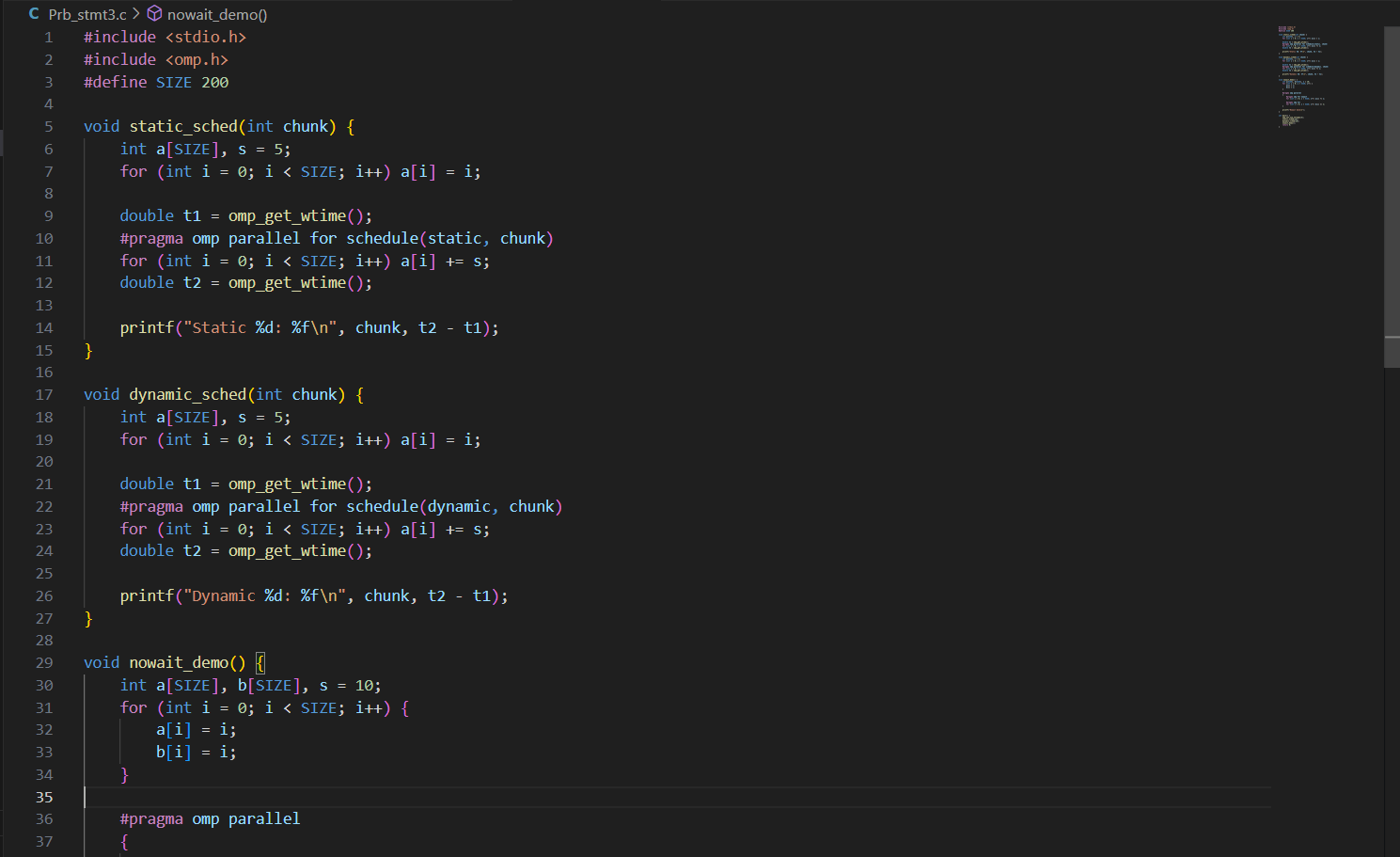
**Analysis:**

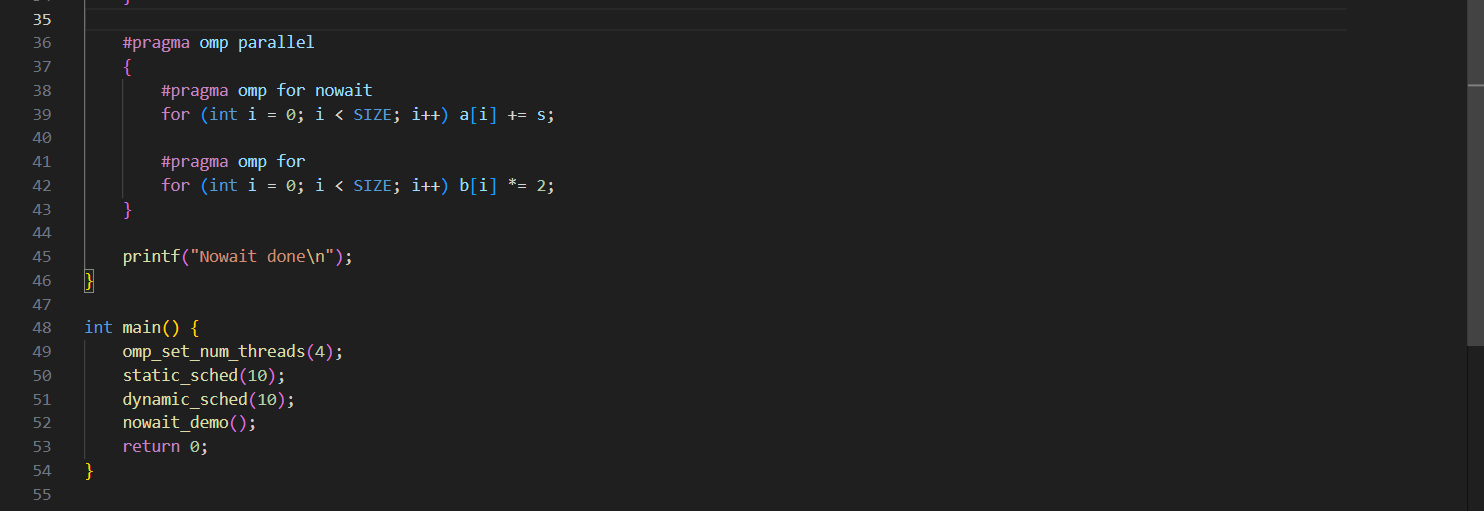
1. For small matrices, there wasn’t much speedup because the work was too little to benefit from threading.
2. For larger matrices, performance improved due to better thread usage.
3. collapse(2) helped by giving OpenMP more pieces of work to balance across threads.
4. Speedup was not perfect because of thread overhead and hardware limits.

**Problem Statement 3:**

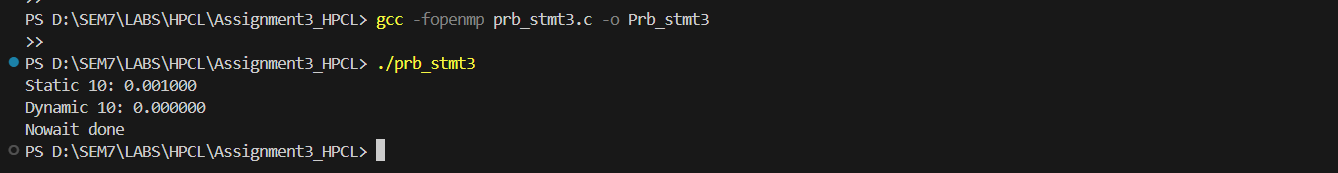
For 1D Vector (size=200) and scalar addition, Write a OpenMP code with the following: i. Use STATIC schedule and set the loop iteration chunk size to various sizes when changing the size of your matrix. Analyze the speedup. ii. Use DYNAMIC schedule and set the loop iteration chunk size to various sizes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

**Screenshots:**

****

****

**Output:**

****

**Information and analysis:**

**OpenMP Concepts Used:**

1. schedule(static):

Divides work into equal parts ahead of time and assigns them to threads.

1. schedule(dynamic):

Assigns work to threads as they finish, good for uneven workloads.

1. nowait:

Lets threads skip waiting at the end of a loop if they don’t need to sync.

**Analysis:**

1. Static scheduling worked well when each task took similar time.
2. Dynamic scheduling was better when some parts of work were slower than others.
3. Choosing the right chunk size mattered — if chunks were too big or too small, performance dropped.
4. Using nowait saved time when threads didn’t need to wait for each other.

**Github Link:**

[**https://github.com/veddnd/HPCprac3**](https://github.com/veddnd/HPCprac3)