**Class:** Final Year (Computer Science and Engineering)

**Year:** 2022-23 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 3**

**Exam Seat No:**

1. 2019BTECS00038 – Sadaf Najeem Mulla

**Title of practical:**

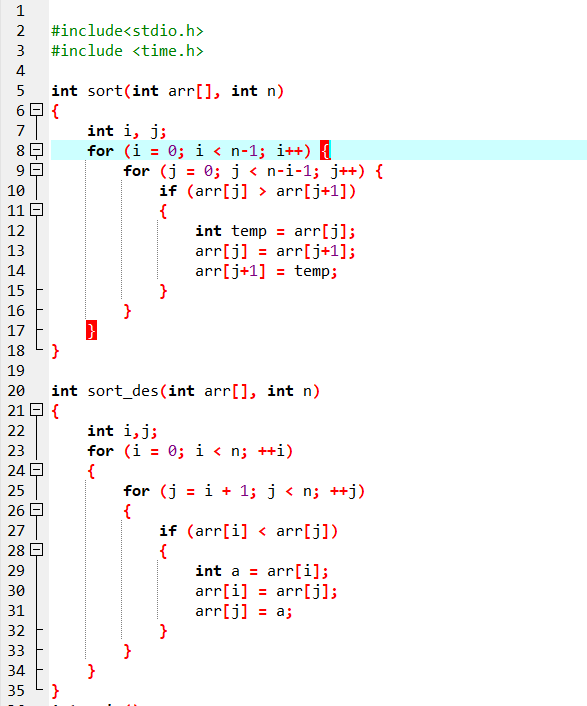
Study and Implementation of schedule, nowait, reduction, ordered and collapse clauses

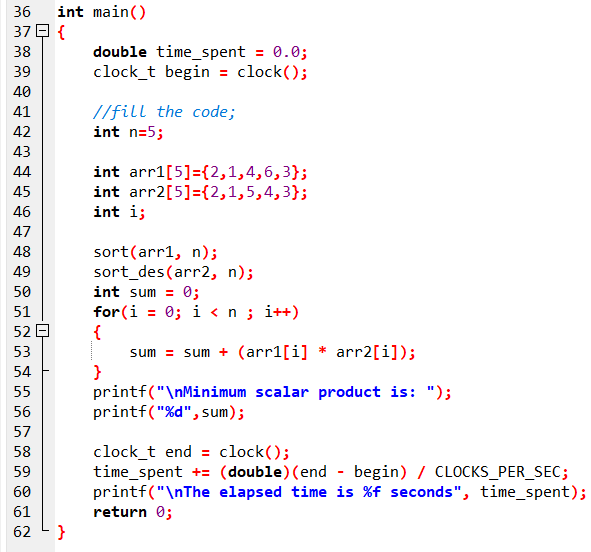
**Problem Statement 1:**

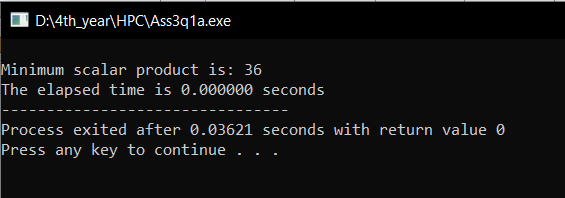
Analyse and implement a Parallel code for below program using OpenMP.

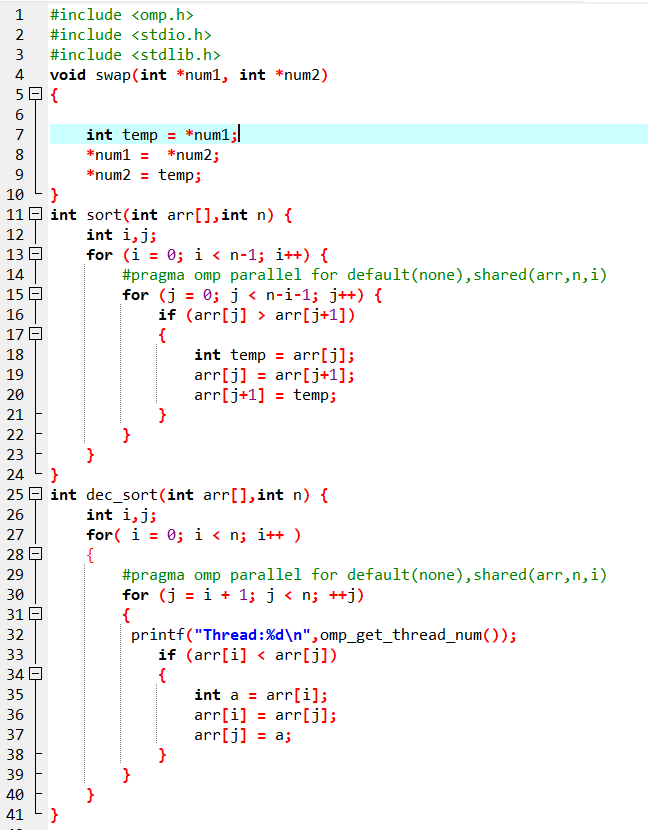
// C Program to find the minimum scalar product of two vectors (dot product)

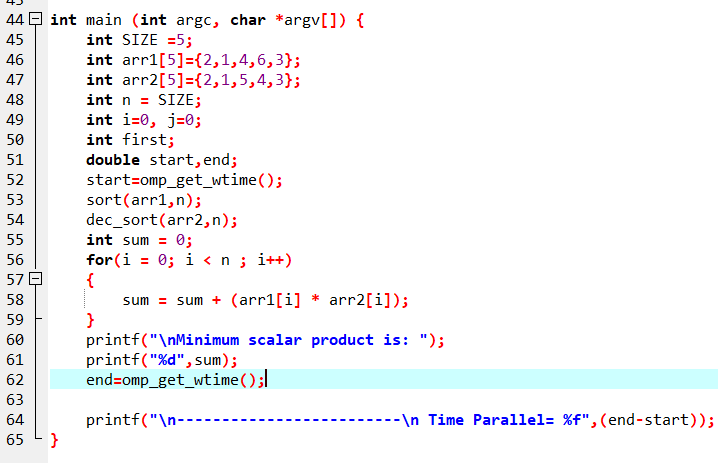
**Screenshots:**

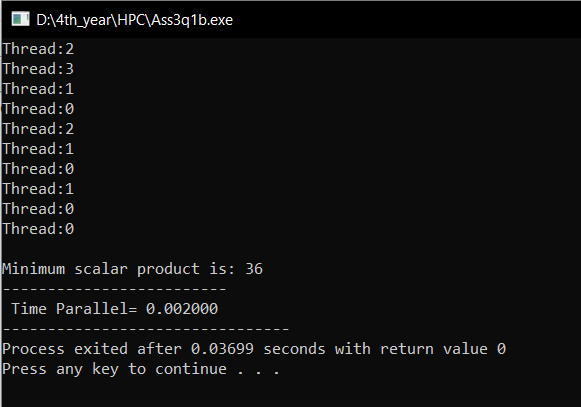












**Information :**

The challenge is to combine the results of all the additions into a single variable. This process is known as a reduction. An improper handling of the reduction process may lead to data races: if all the threads attempt to read from and write to the same memory location they might overwrite each other's results and arrive at the wrong answer. One way to handle this situation is to use what's called a critical section. The code in a critical section can only be executed by a single thread at any given point in time. We can use a critical section to prevent different threads from accessing the same memory location in an unsafe manner, and thereby avoid data races.

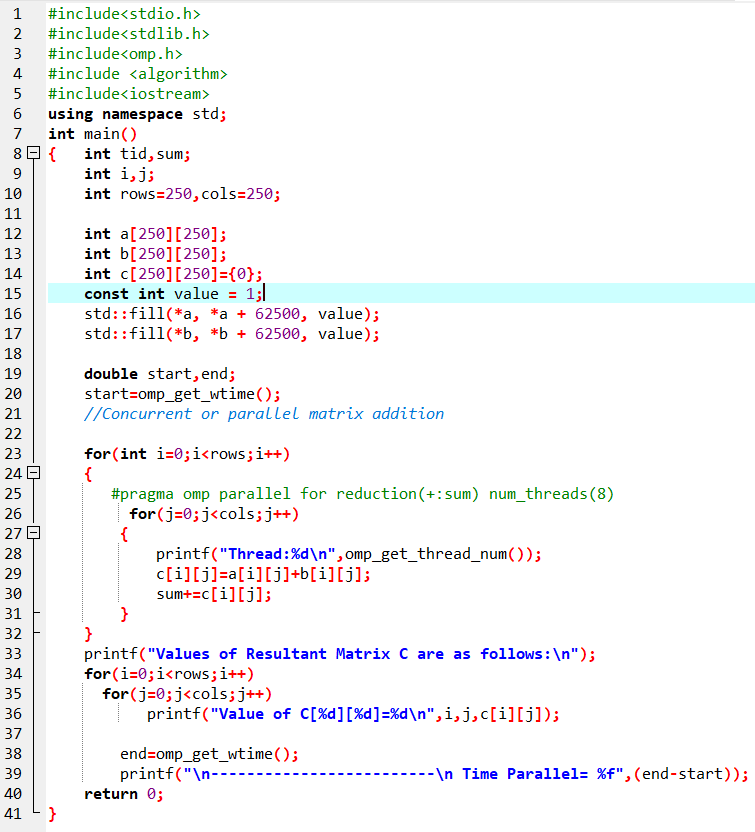
**Problem Statement 2:**

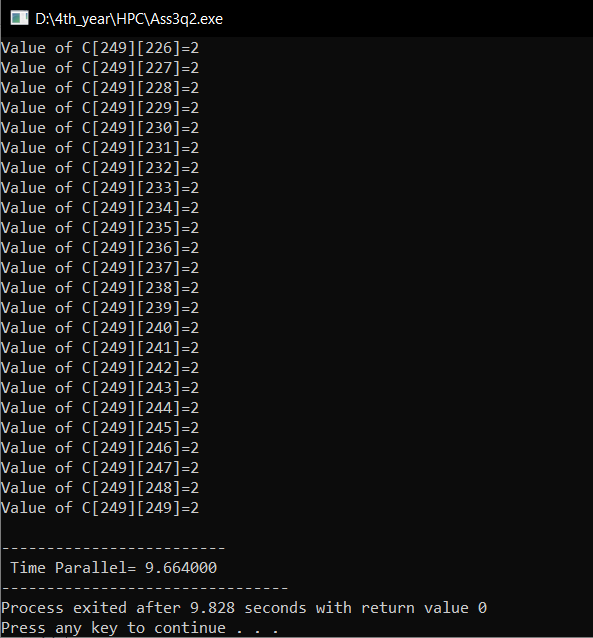
Write OpenMP code for two 2D Matrix addition, vary the size of your matrices from 250, 500, 750, 1000, and 2000 and measure the runtime with one thread (Use functions in C in calculate the execution time or use GPROF)

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:**





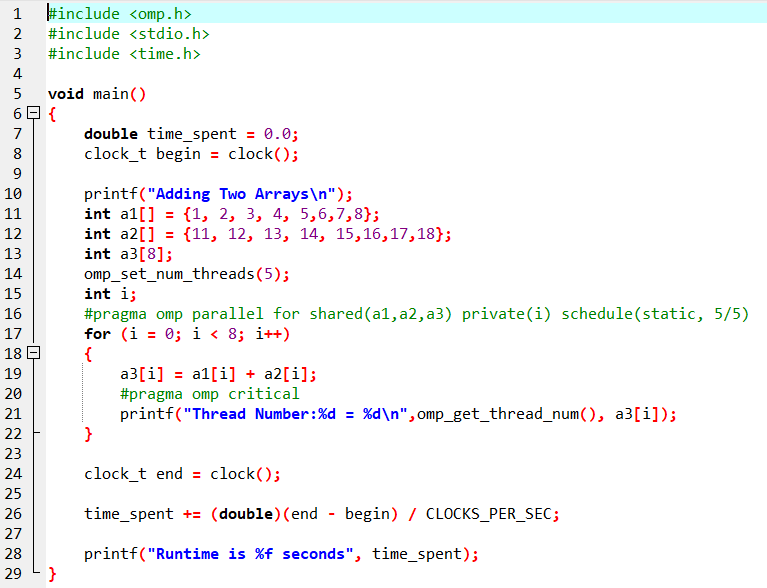
**Information :**

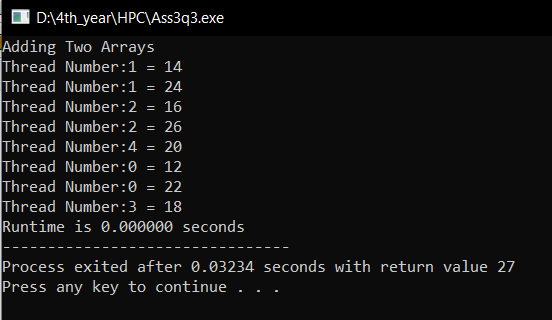
The parallel code is slightly faster but still much slower than the sequential version so we use reduction.An improper handling of the reduction process may lead to data races: if all the threads attempt to read from and write to the same memory location they might overwrite each other's results and arrive at the wrong answer. One way to handle this situation is to use what's called a critical section. The code in a critical section can only be executed by a single thread at any given point in time. We can use a critical section to prevent different threads from accessing the same memory location in an unsafe manner, and thereby avoid data races.

**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 :**

The shared keyword indicates that the arrays are shared in the same memory space for all threads, and the private keyword indicates that each thread will have its own copy of the index counter i that it will increment.

The schedule keyword is used in this pragma to indicate how many consecutive iterations of the loop, and thus computations on consecutive elements of the arrays, that each thread will execute.

**Github Link:** [**https://github.com/sadafmulla/HPC\_LAB/tree/main/Assignment3**](https://github.com/sadafmulla/HPC_LAB/tree/main/Assignment3)