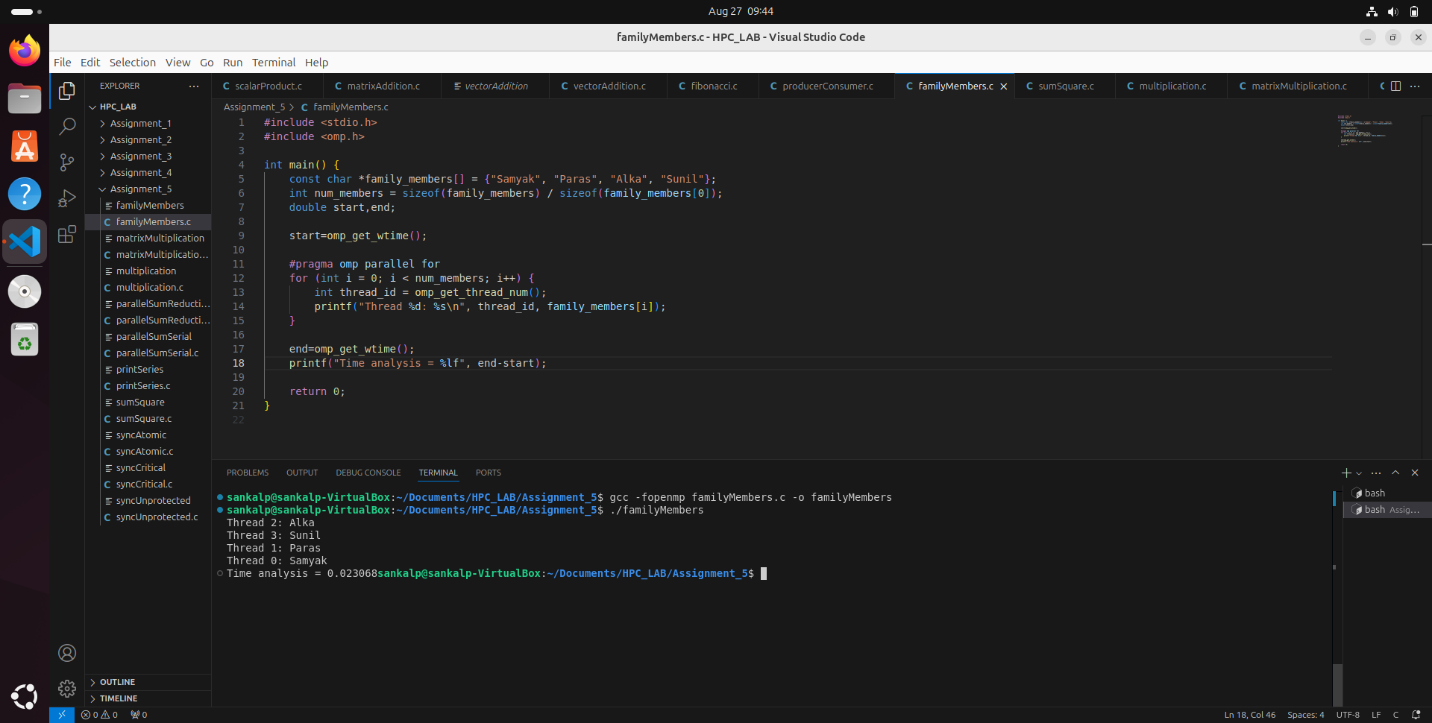
**Practical No. 4: Study and Implementation of Synchronization**

**Class**: Final Year (Computer Science and Engineering)  
**Year**: 2024-25  
**Semester**: 1  
**Course**: High Performance Computing Lab  
**Exam Seat No**: 21510072  
**Title of Practical**: Study and Implementation of Synchronization

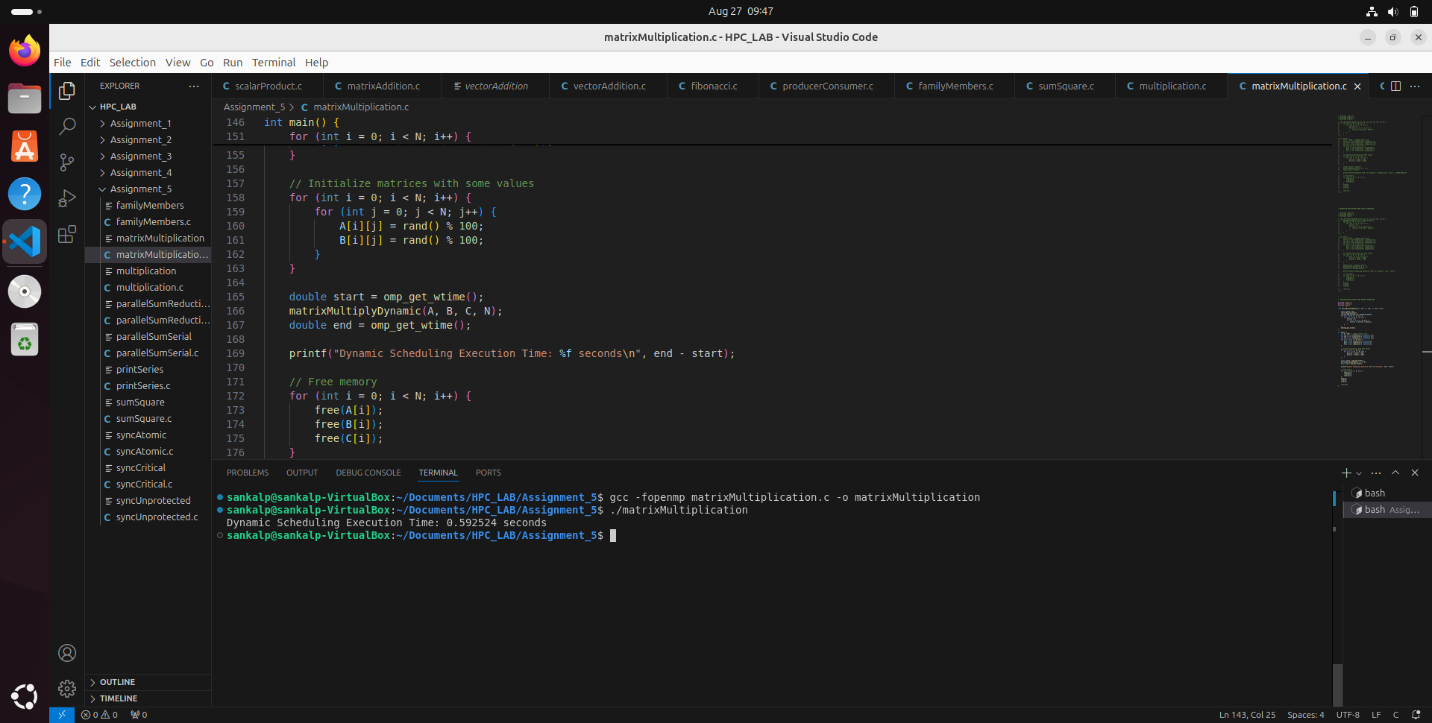
**Problem Statement 1: Parallel Printing of Family Member Names and Job IDs**

**Analysis**:  
This problem involves printing the names of your family members along with their respective job IDs using different threads. The OpenMP directives ensure that each thread prints one name and its associated job ID. This exercise helps you understand the basics of task distribution across multiple threads in a parallel environment.

**Screenshots**:  


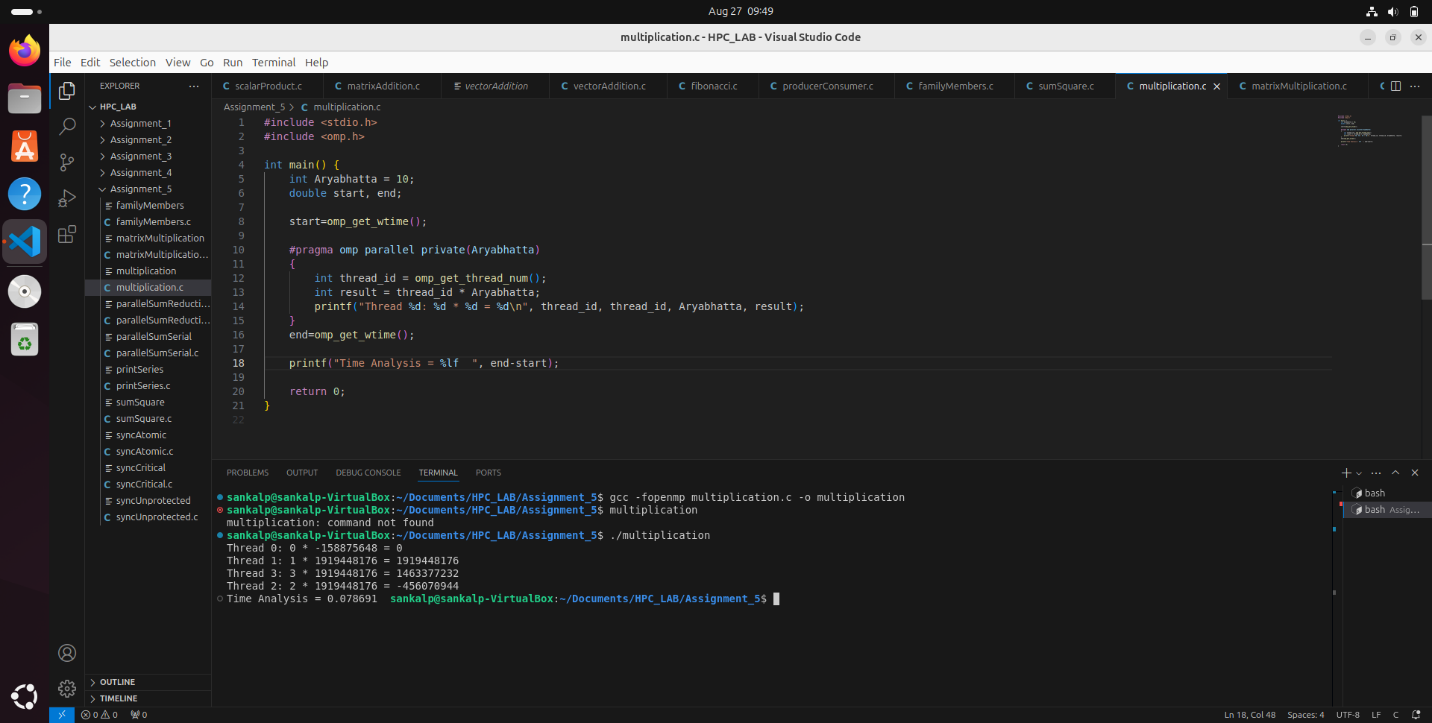
**Problem Statement 2: Sum of Squares of Thread IDs**

**Analysis**:  
This problem requires each thread to calculate the square of its own thread ID and add it to a shared sum. The challenge lies in efficiently distributing the computation across threads and aggregating the results without causing race conditions.

**Screenshots**:  


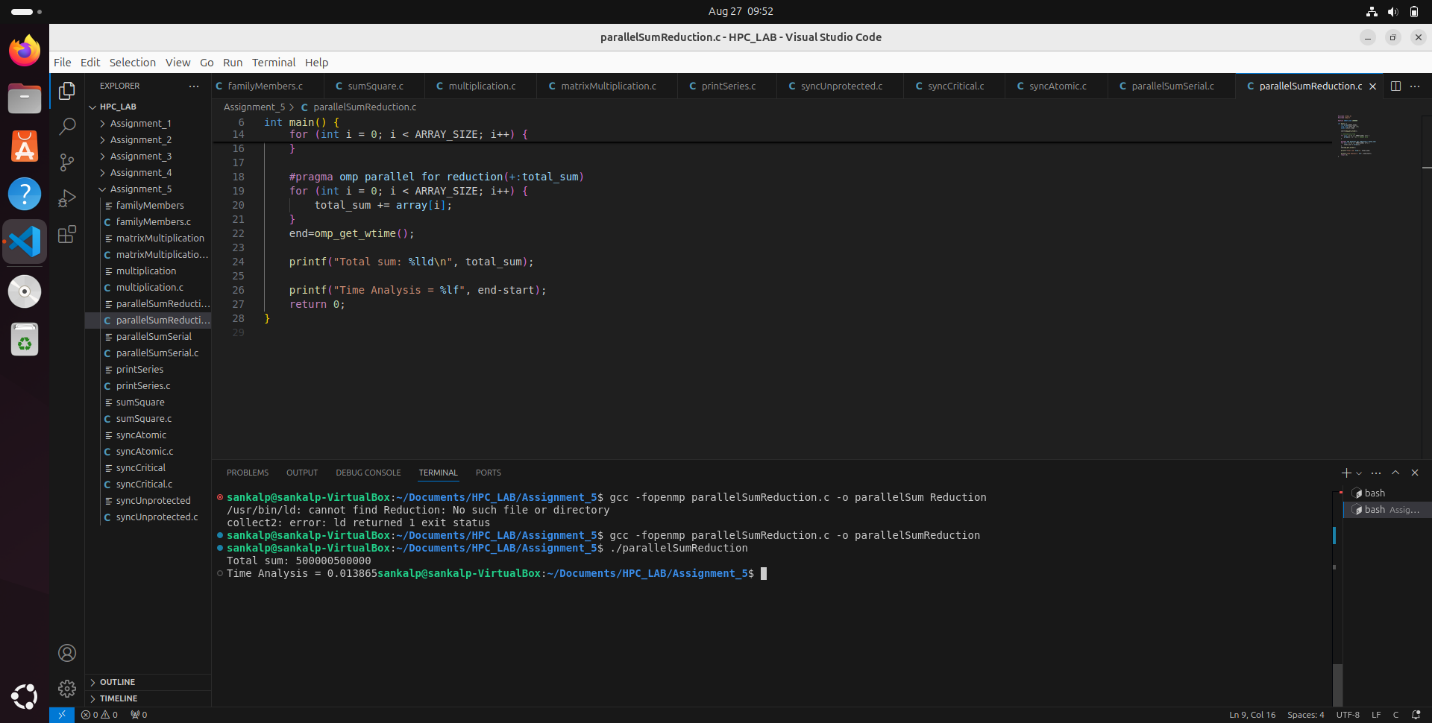
**Problem Statement 3: Multiplication of Thread ID with Private Variable**

**Analysis**:  
This problem demonstrates the use of a private variable in OpenMP, where each thread multiplies its ID with the private variable Aryabhatta. The key concept is ensuring that each thread works with its own copy of the variable to avoid race conditions and achieve correct results.

**Screenshots**:  


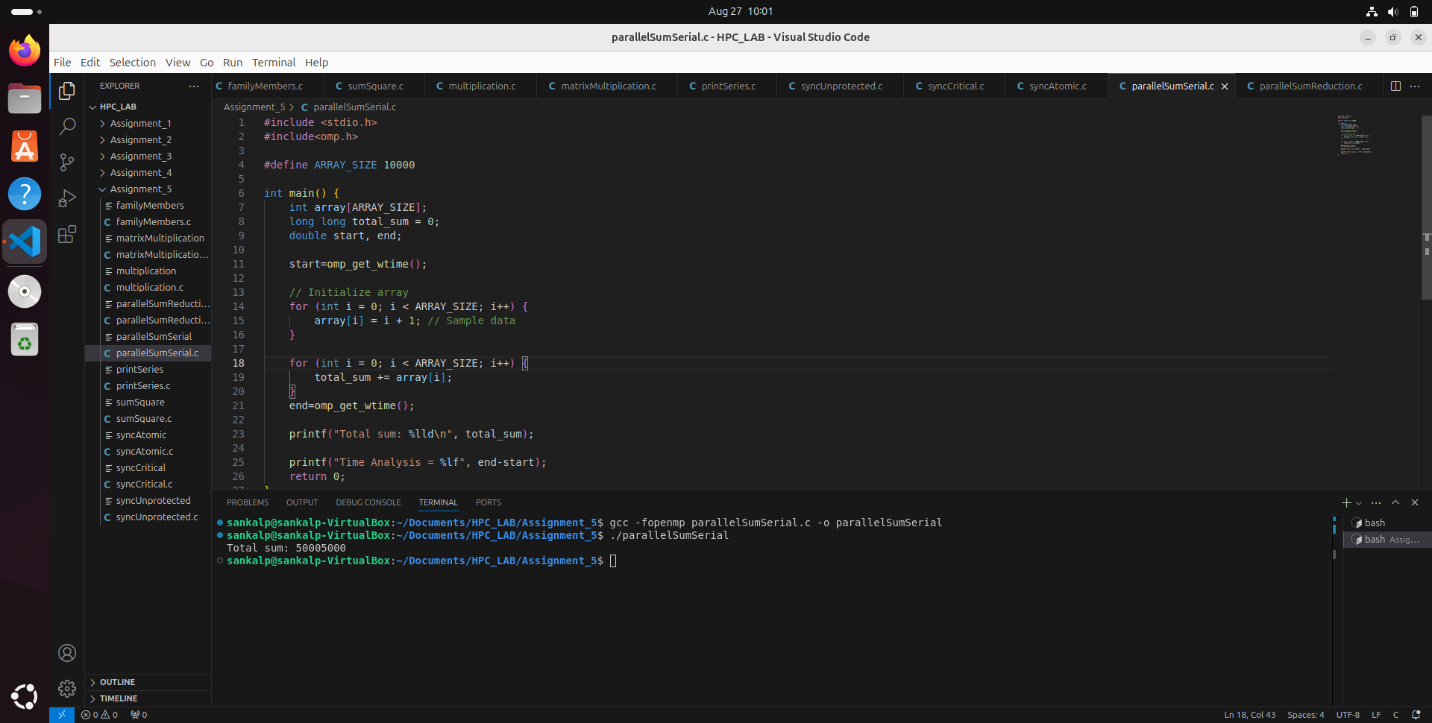
**Problem Statement 4: Partial Sum of First 20 Natural Numbers**

**Analysis**:  
The goal here is to calculate the sum of the first 20 natural numbers using parallelism. Each thread computes a portion of the sum, and the lastprivate clause ensures that the final sum is correctly computed and available outside the parallel region.

**Screenshots**:  


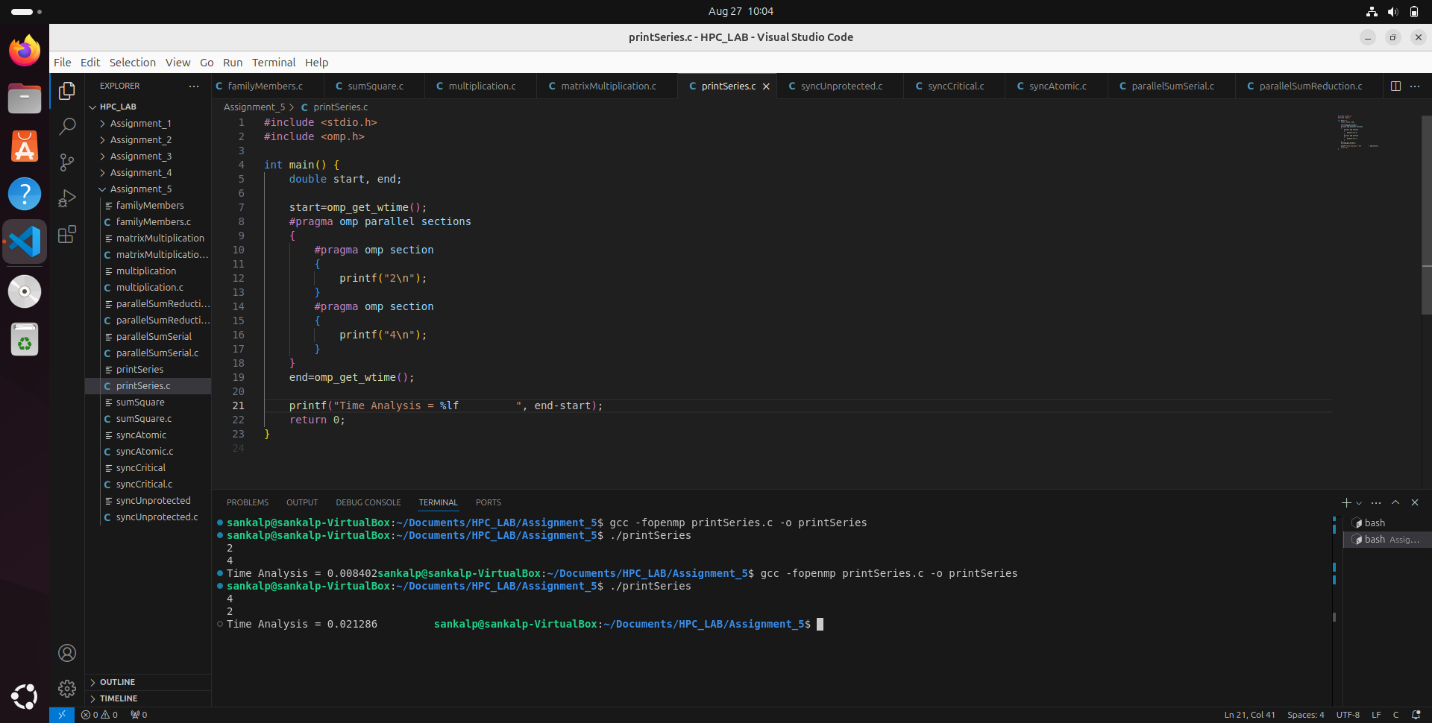
**Problem Statement 5: Parallel Matrix Multiplication with Static and Dynamic Scheduling**

**Analysis**:  
This problem explores the performance differences between static and dynamic scheduling in OpenMP for parallel matrix multiplication. Static scheduling divides work evenly among threads at compile time, while dynamic scheduling allocates tasks to threads at runtime, potentially leading to better load balancing.

**Screenshots**:  


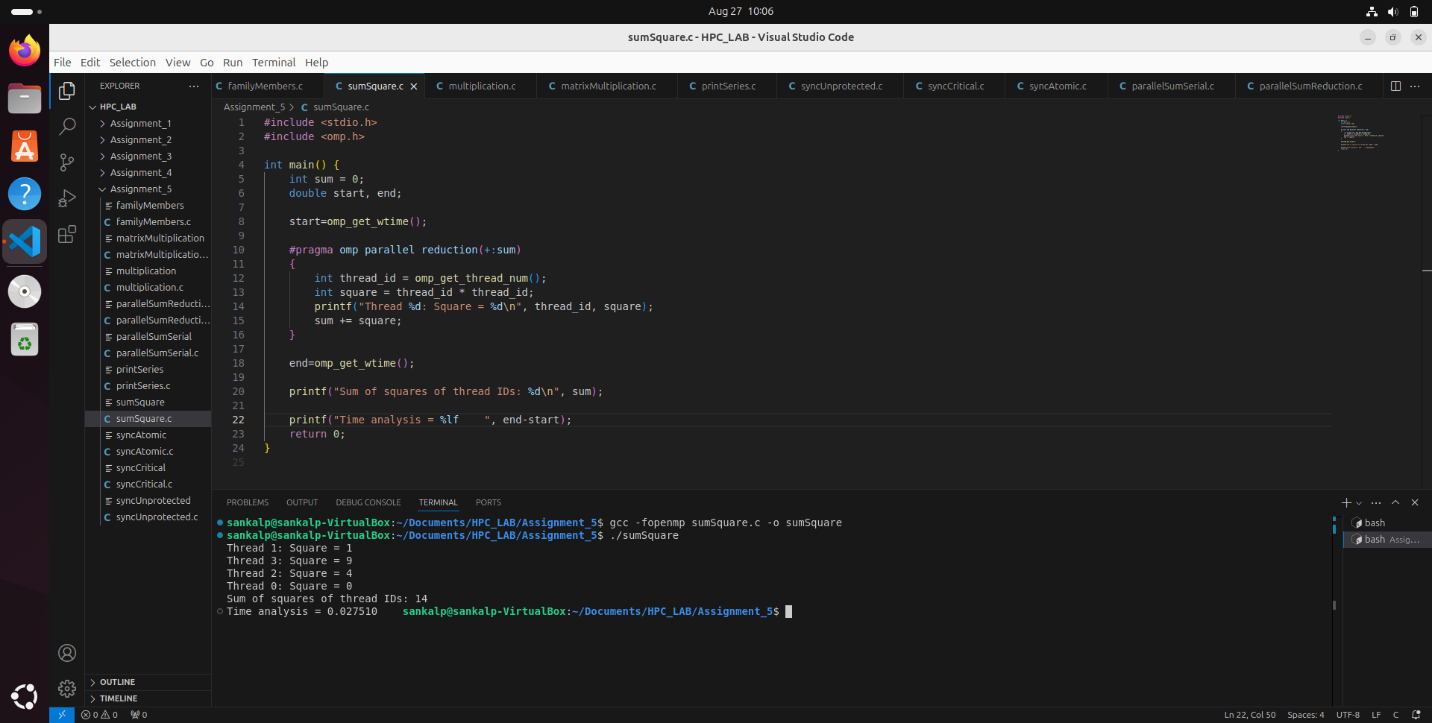
**Problem Statement 6: Parallel Series Printing of 2 and 4**

**Analysis**:  
This problem involves printing two independent series (2 and 4) concurrently using different threads. It demonstrates basic parallelism by assigning each series to a separate thread, ensuring that both series are printed simultaneously without interference.

**Screenshots**:  


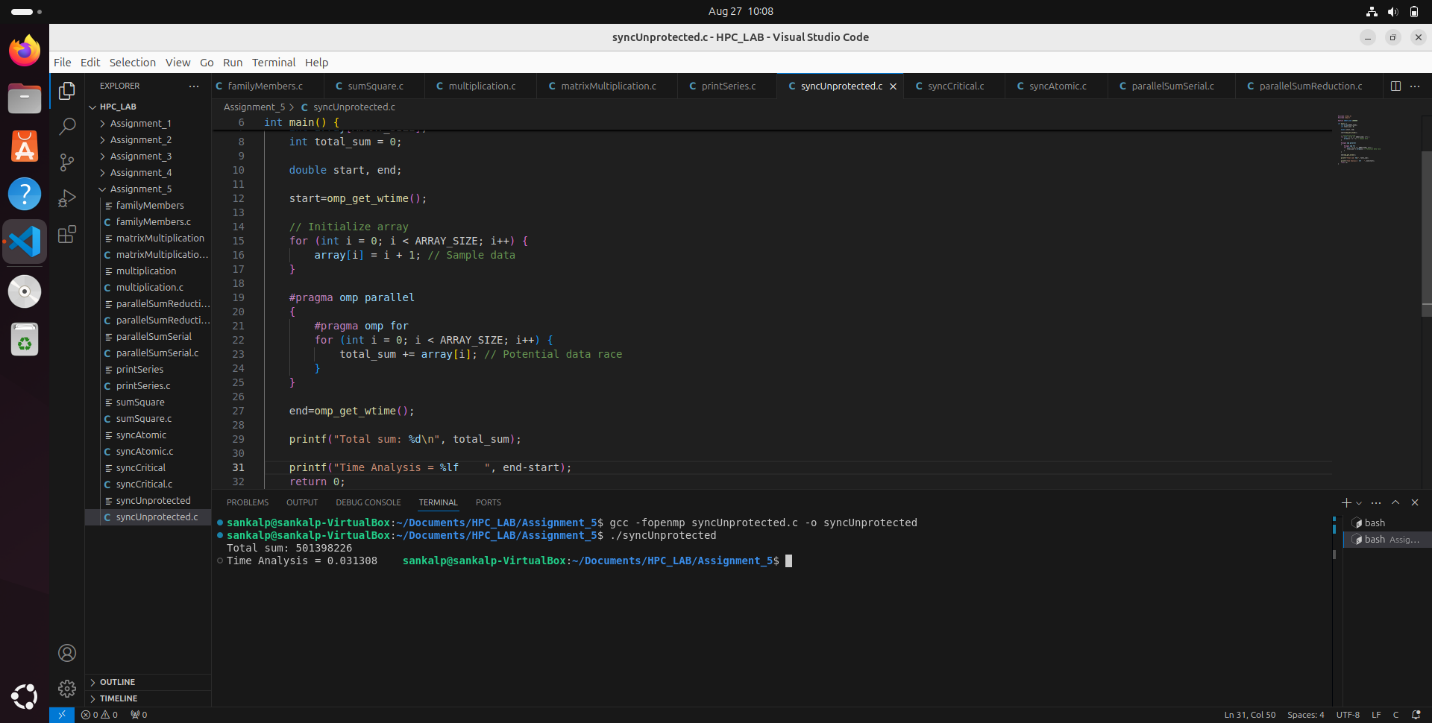
**Problem Statement 7: Concurrent Update of Shared Variable with Synchronization**

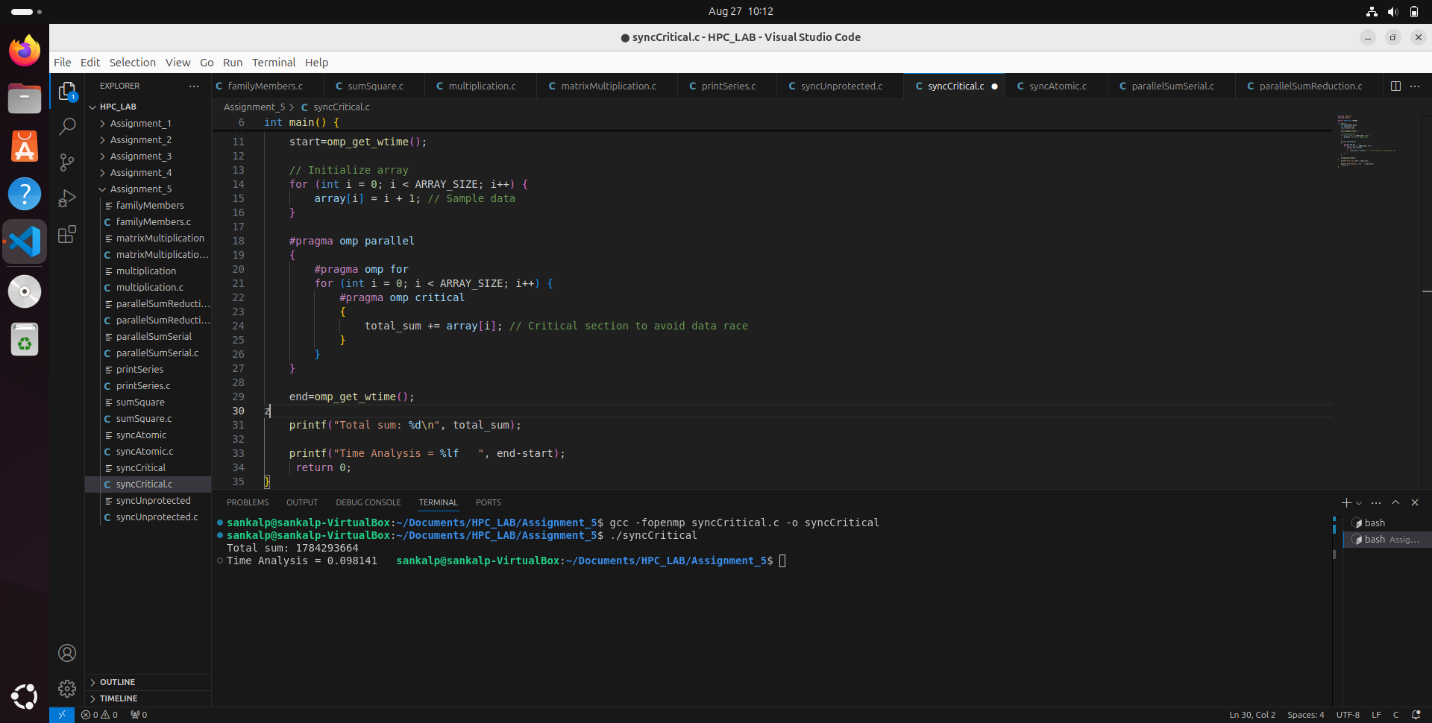
**Analysis**:  
This problem addresses the issue of race conditions by updating a shared variable total\_sum concurrently using multiple threads. The use of critical or atomic directives in OpenMP ensures synchronized access to the shared variable, preventing incorrect results due to race conditions.

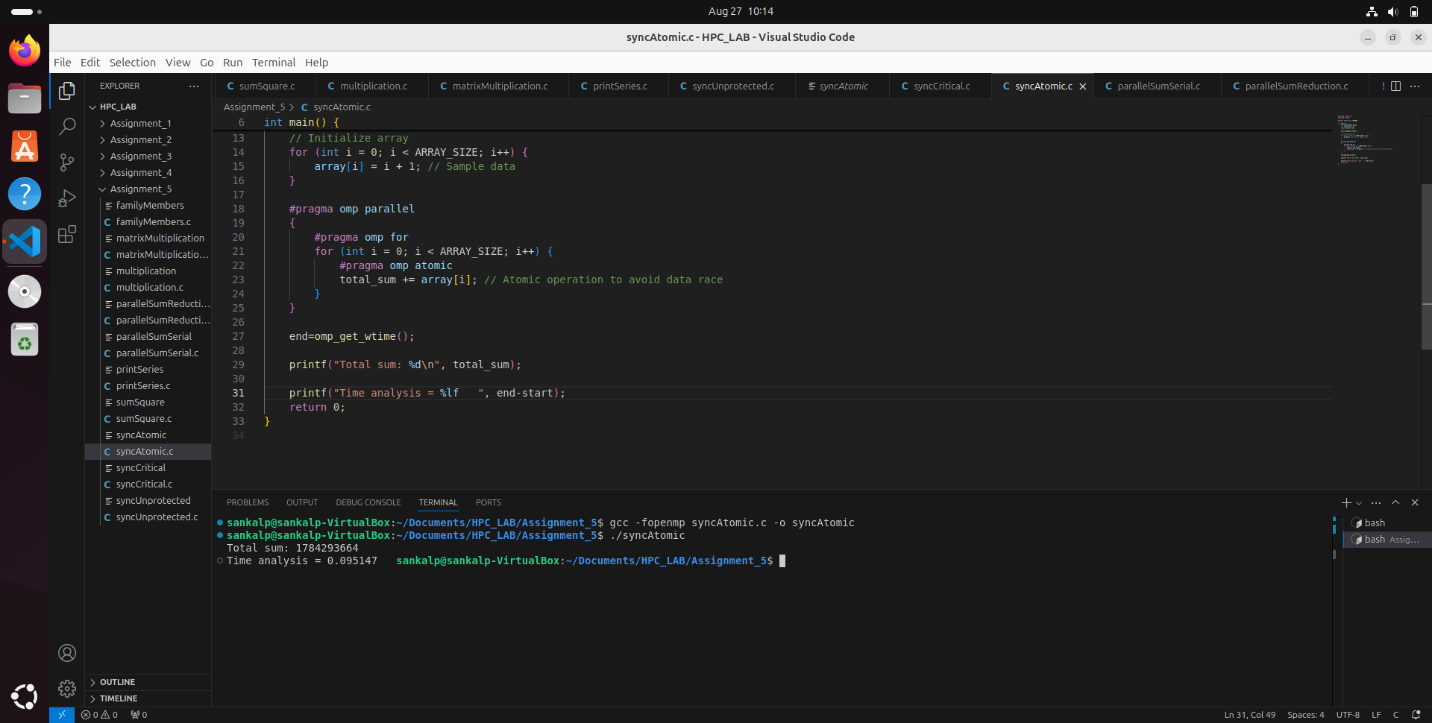
**Screenshots**:  


**Problem Statement 8: Parallel Summation with Reduction Clause**

**Analysis**:  
The task is to sum all elements of a large array in parallel using the reduction clause in OpenMP. This clause ensures that the sum is computed efficiently across multiple threads without the need for manual synchronization, making the program scalable and efficient for large data sets.

**Screenshots**:  






Github Link: https://github.com/sankalp56/HPC\_Assignments.git