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

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

**Course:** High Performance Computing Lab

**Practical No. 4**

**Exam Seat No:**

**Title of practical:**

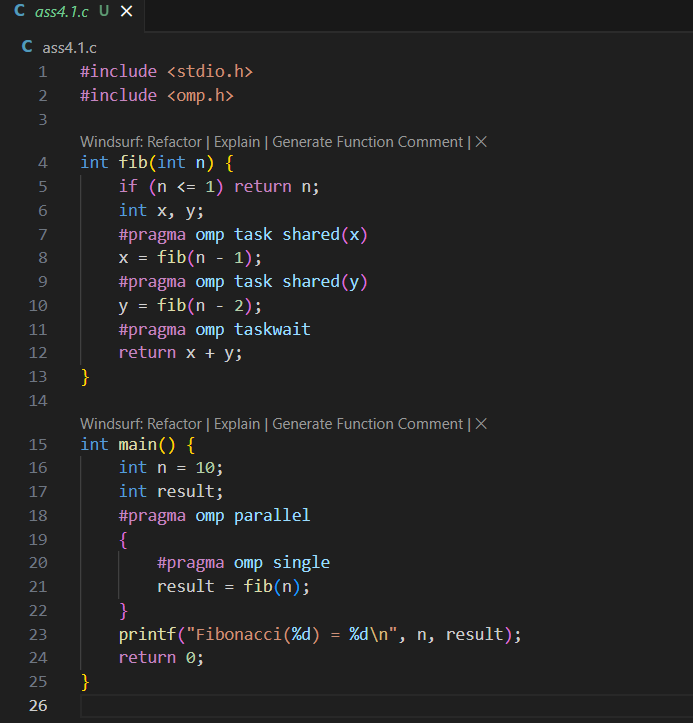
Study and Implementation of Synchronization

**Problem Statement 1:**

# Analyze and implement a Parallel code for below programs using OpenMP considering synchronization requirements. (Demonstrate the use of different clauses and constructs wherever applicable)

# Fibonacci Computation:

**Screenshots:**

****

****

**Information:**

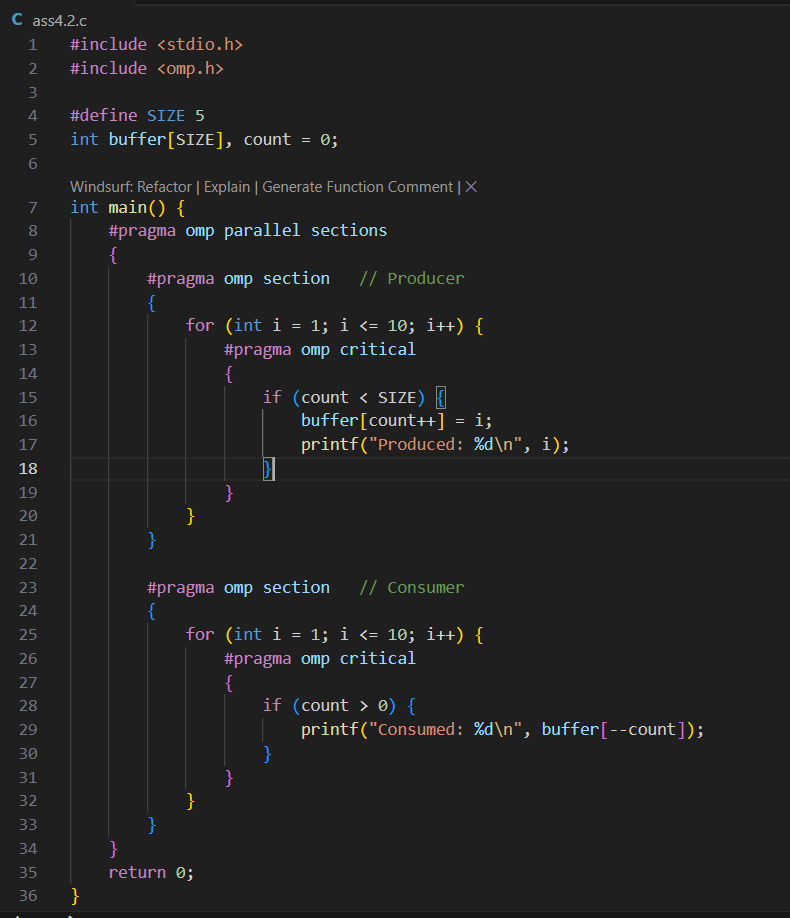
* **Uses #pragma omp task to create parallel tasks for recursive calls.**
* **Uses #pragma omp taskwait to synchronize tasks.**

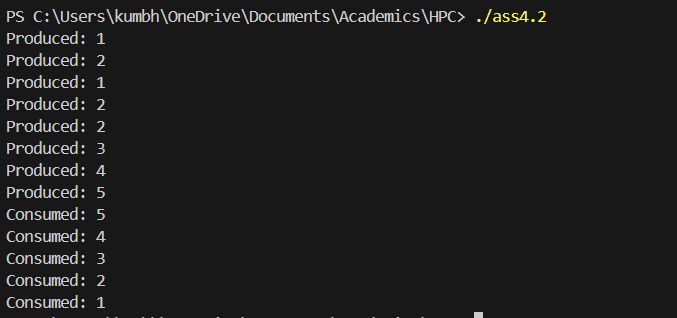
**Problem Statement 2:**

# Analyze and implement a Parallel code for below programs using OpenMP considering synchronization requirements. (Demonstrate the use of different clauses and constructs wherever applicable)

## Producer Consumer Problem

**Screenshots:**

****

****

**Information:**

* **Uses #pragma omp parallel sections to run producer and consumer in parallel.**
* **Uses #pragma omp critical to protect access to the shared buffer and count variable.**
* Demonstrates synchronization using critical sections to avoid race conditions.
* Suitable for small-scale producer-consumer problems; for high performance, more advanced synchronization (like locks or atomic operations) may be needed.

**Analysis:**

* critical sections as a basic but effective way to ensure synchronization and prevent race conditions when multiple threads access shared resources. The document notes that while this approach is suitable for small-scale problems, for higher performance in more complex scenarios, more advanced synchronization mechanisms such as
* **locks** or **atomic operations** may be necessary

**Github Link: https://github.com/Anjali1874/HPC-Lab**