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

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

**Course:** High Performance Computing Lab

**Practical No. 4**

**Exam Seat No: 22510092**

**Title of practical:**

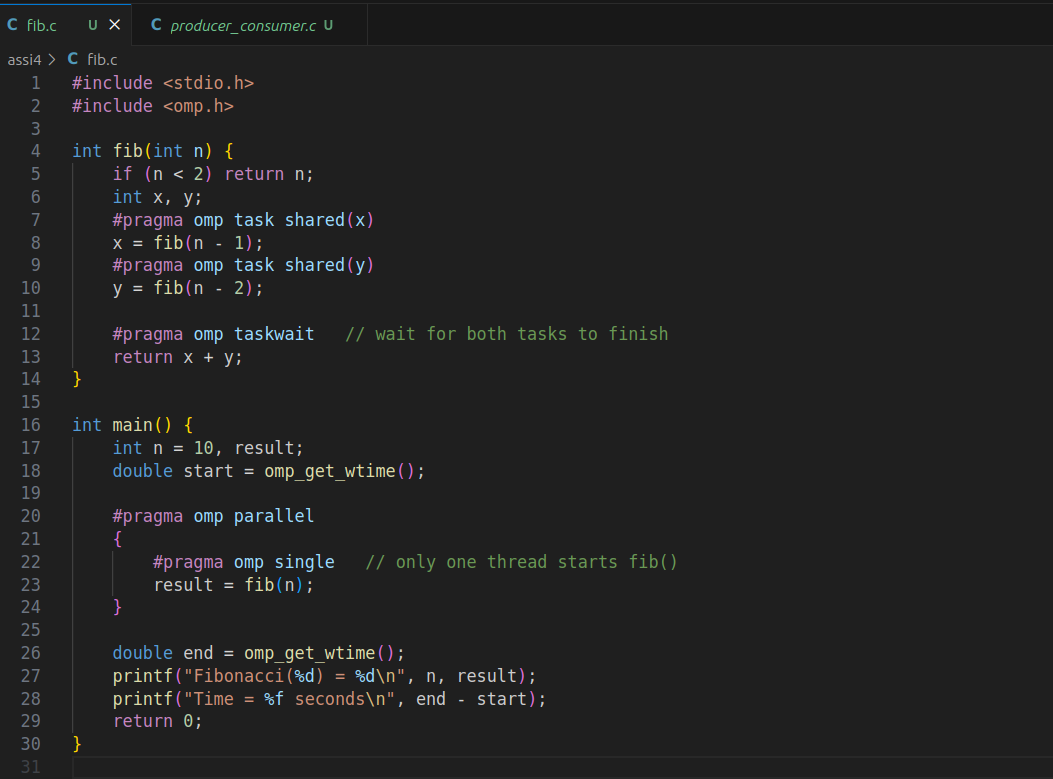
Study and Implementation of Synchronization

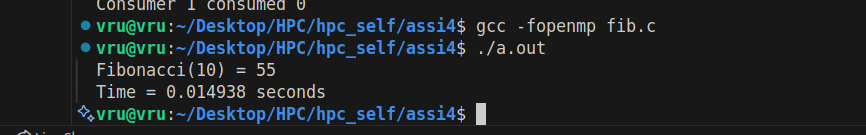
**Problem Statement 1:**

# Analyse 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:**

* #pragma omp parallel: Starts parallel region with multiple threads.
* #pragma omp single: Only one thread starts Fibonacci, but subtasks are executed by other threads.
* #pragma omp task: Creates independent tasks for recursive calls.
* #pragma omp taskwait: Synchronizes tasks (wait until both subtasks finish before returning).

### Analysis:

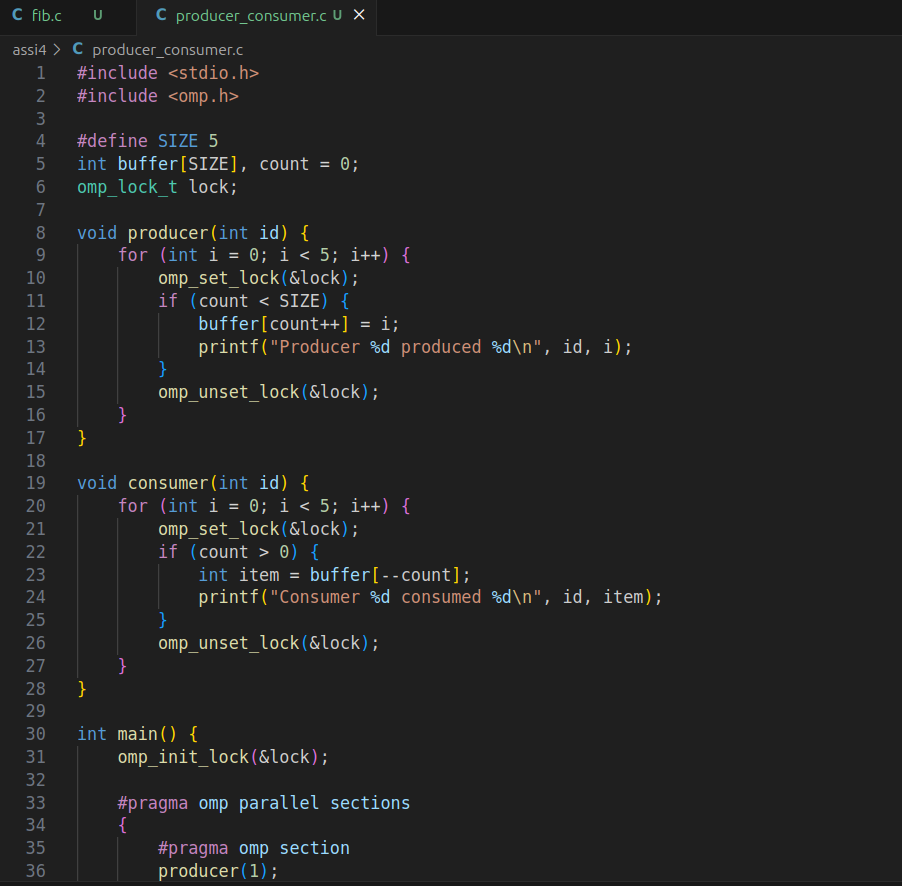
* Shows **task parallelism** with synchronization.
* Multiple Fibonacci calls are executed **concurrently**, improving performance for large n.
* **Synchronization** is essential (taskwait), else results would be wrong.
* Demonstrates how **dependencies** in recursive problems can be parallelized safely.

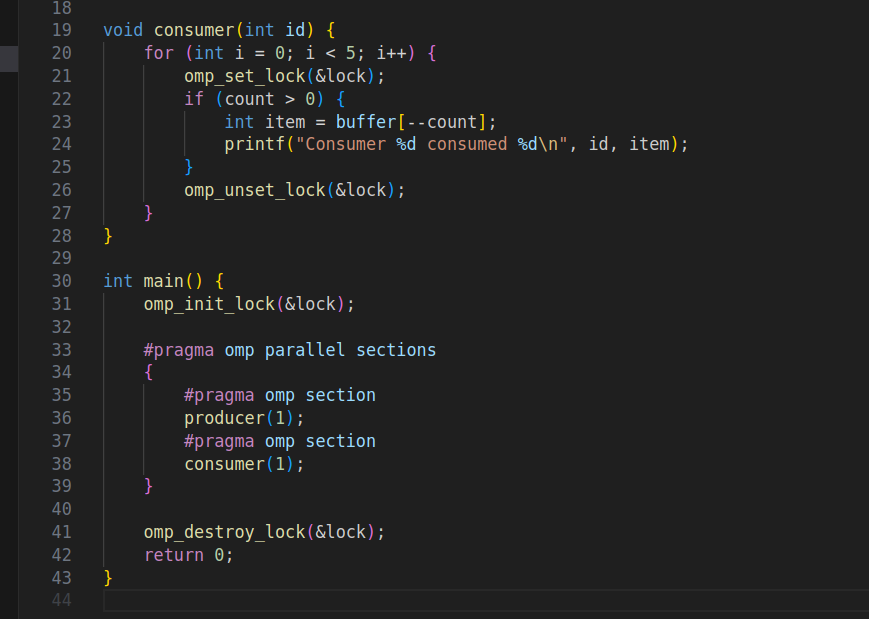
**Problem Statement 2:**

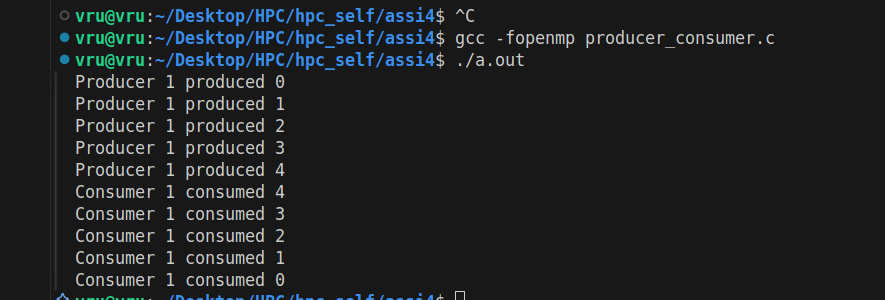
# Analyse and implement a Parallel code for below programs using OpenMP considering synchronization requirements. (Demonstrate the use of different clauses and constructs wherever applicable)

## Producer Consumer Problem

**Screenshots:**

****

****

****

**Information:**

### **Information:**

* #pragma omp parallel sections: Creates producer and consumer threads.
* #pragma omp critical: Ensures **mutual exclusion** (only one thread updates buffer at a time).
* Shared variables: buffer[] and count.
* Prevents **race conditions** while producing/consuming.

### Analysis:

* Demonstrates **synchronization** in shared buffer access.
* Producer adds items; consumer removes them.
* Without critical, both threads could **corrupt buffer** or give wrong counts.
* Shows real-world **producer–consumer model** with OpenMP synchronization.

**Github Link:**

https://github.com/Vru01/HPC\_22510092