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

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

**Course:** High Performance Computing Lab

**Practical No. 4**

**Name: Aadarsh V. Nandedkar**

**Exam Seat No: 22510047**

**Batch: B-8**

**Title of practical:**

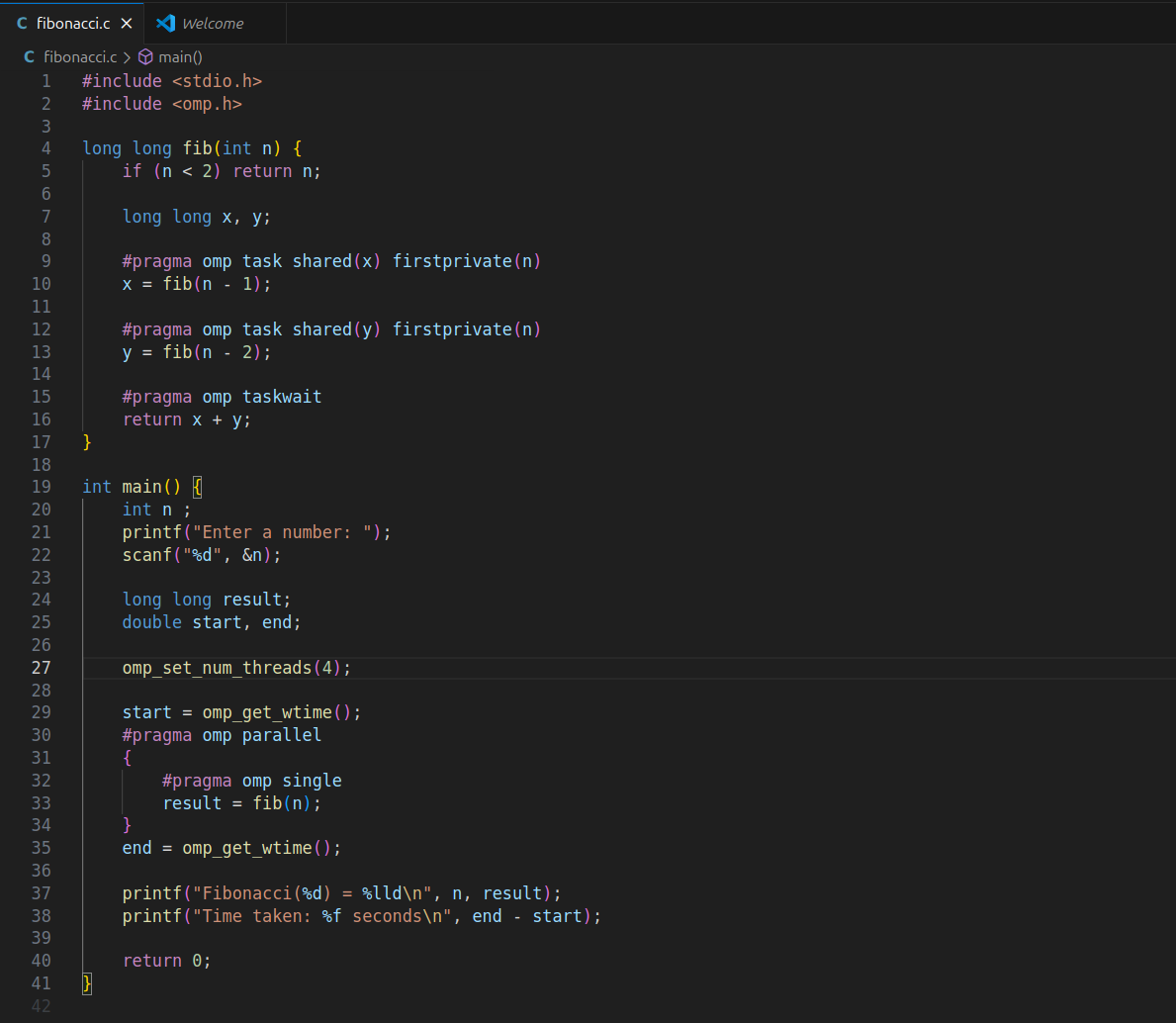
Study and Implementation of Synchronization

**Problem Statement 1:**

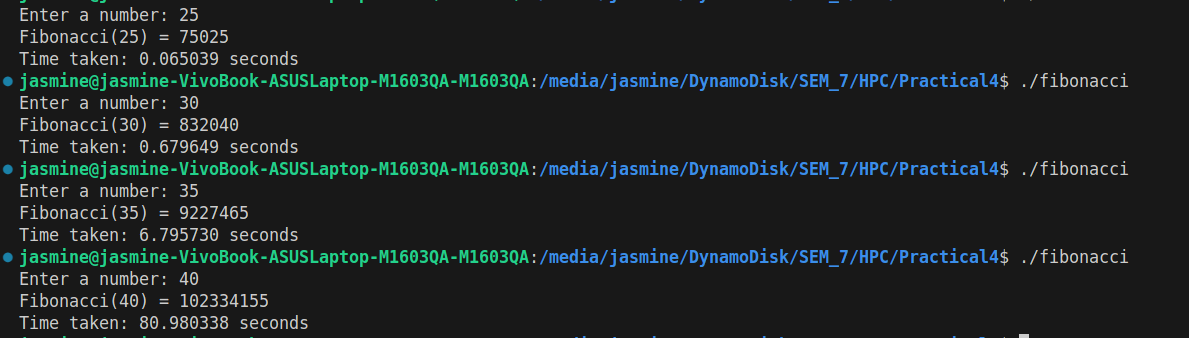
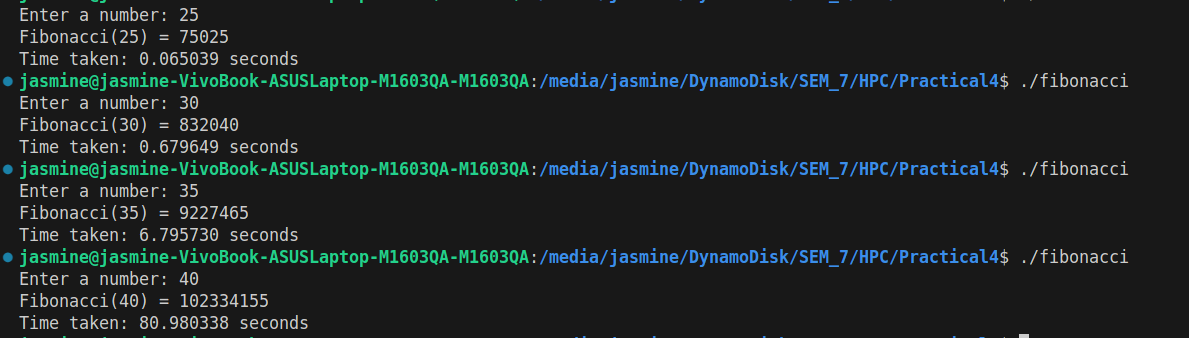
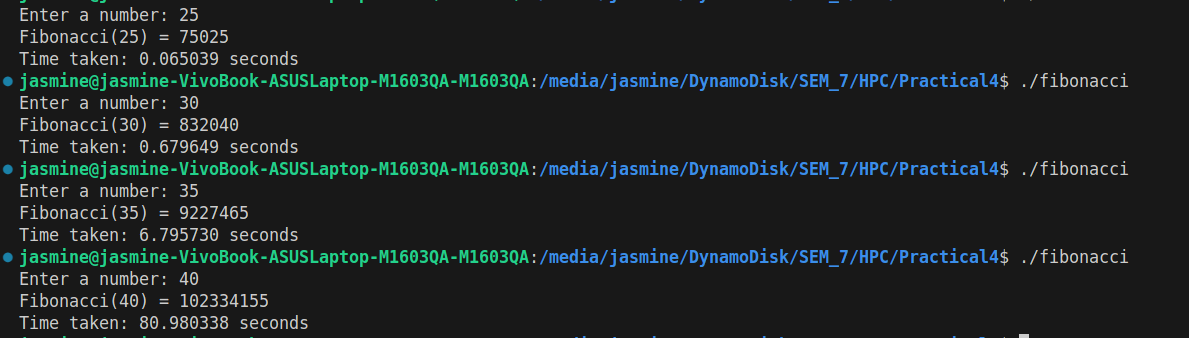
# Analyze and implement a Parallel code for below programs using OpenMP considering synchronization requirements. (Demonstrate the use of different clauses and constructs wherever applicable)

# Fibonacci Computation:

**Screenshots:**

****

**OutPut:**

****

**Information:**

### #pragma omp single

### Ensures only one thread in the team starts the fib(n) call. Other threads wait for tasks to appear and help execute them

### #pragma omp task

Creates an independent unit of work that can be executed by any available thread.

**shared(x)** → All threads see the same x variable (final result will be stored there).

**firstprivate(n)** → Each task gets its own **private copy** of n with the current value from the calling function.

### #pragma omp taskwait

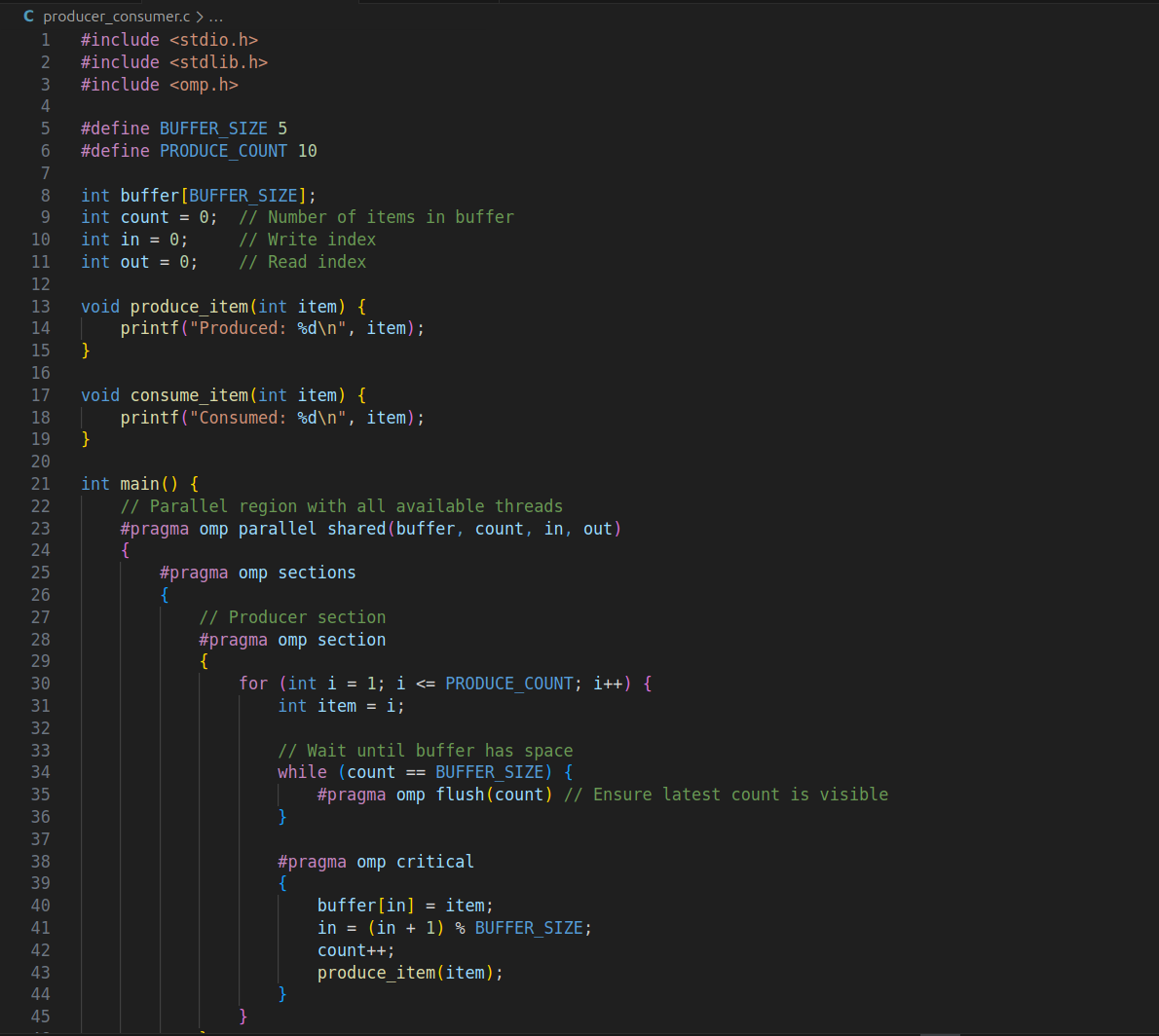
Ensures **both tasks finish** before continuing.  
Without it, the function could return before x and y are computed.

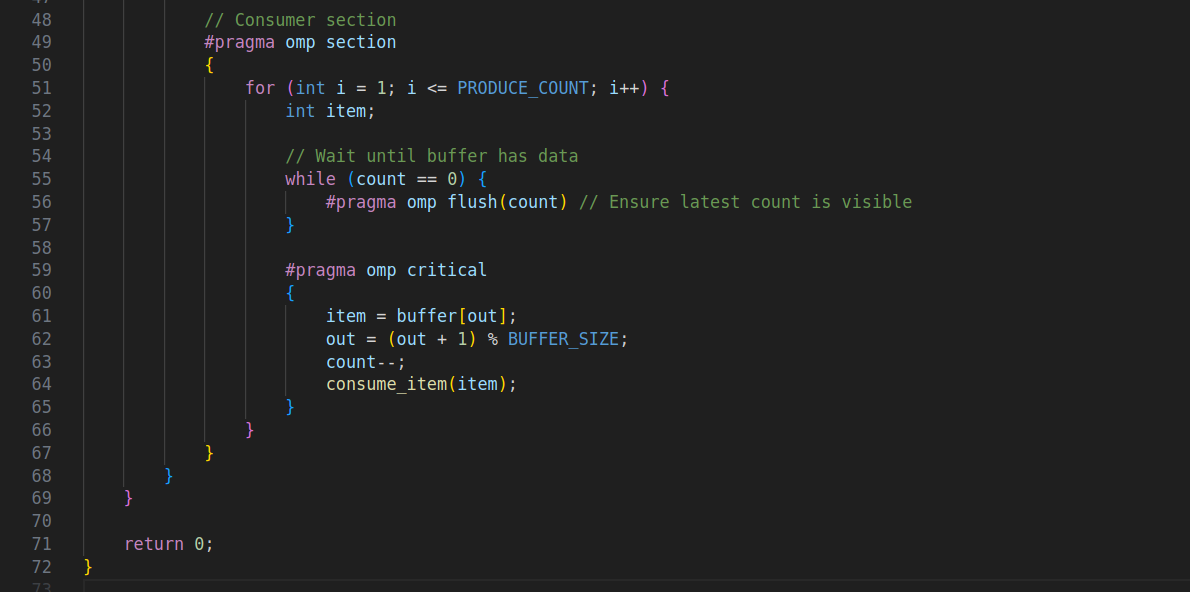
**Problem Statement 2:**

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

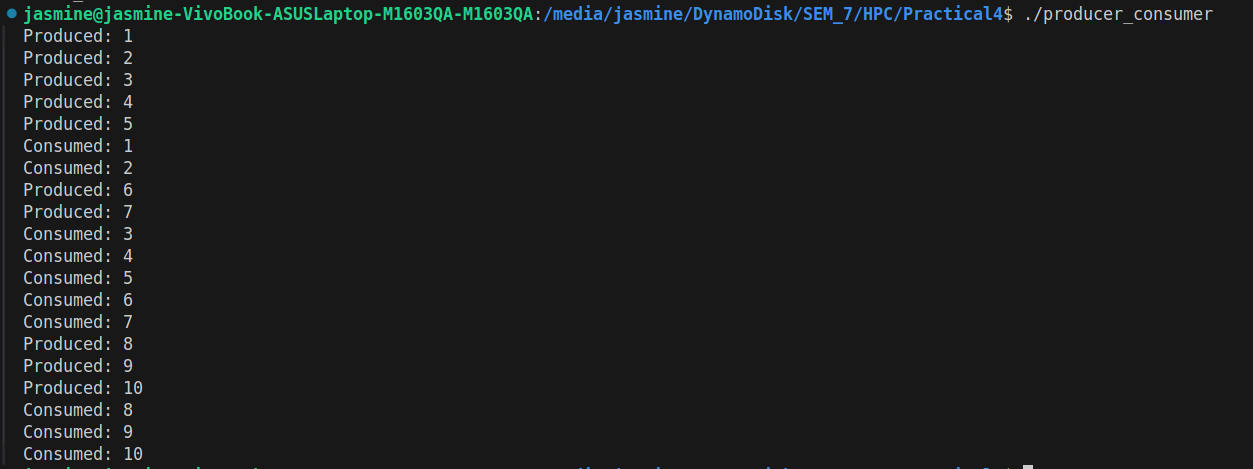
## Producer Consumer Problem

**Screenshots:**

****

****

**Output:**

****

**Information:**

**parallel →** create a team of threads.

**sections / section →** split producer and consumer.

**critical →** prevent race conditions.

**flush →** ensure memory visibility between threads.

**shared →** share buffer, counters, and indices across threads.

**Github Link:**

<https://github.com/av-nandedkar/HPC>