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

**Year:** 2024-25 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 4**

**Exam Seat No:**

**Title of practical:**

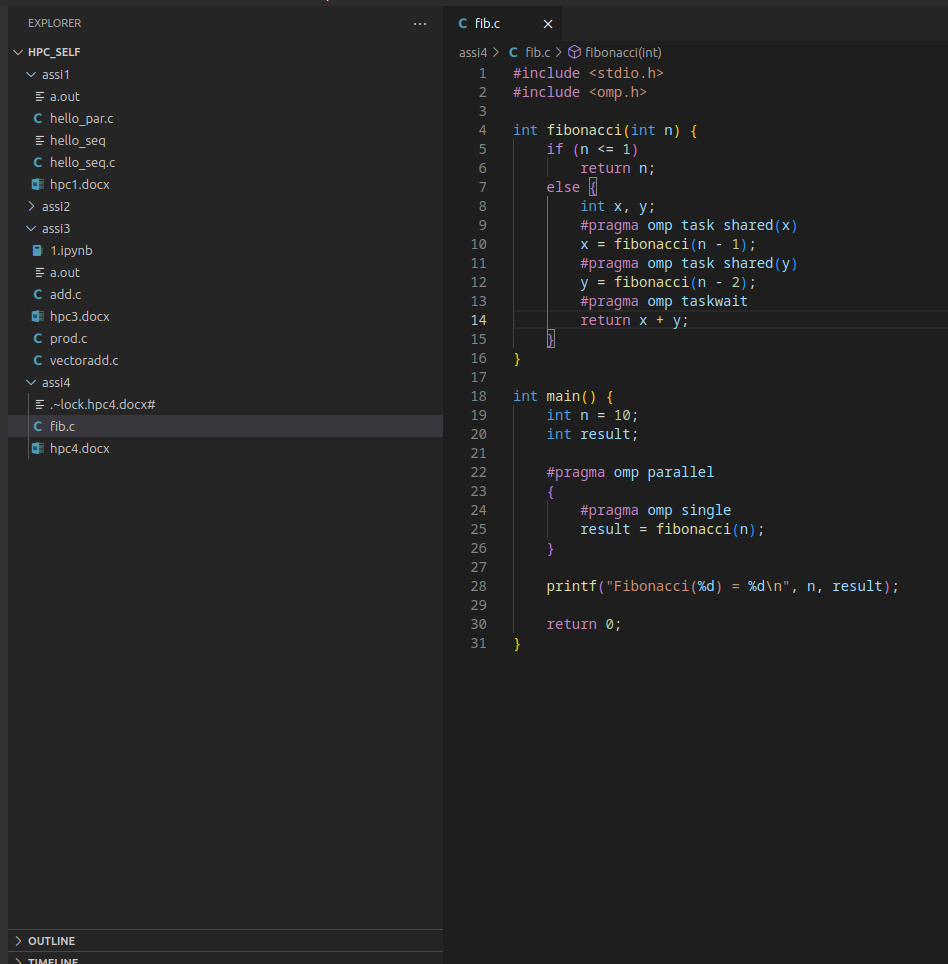
Study and Implementation of Synchronization

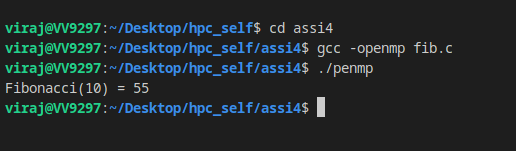
**Problem Statement 1:**

# 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:**

**1. Recursive Fibonacci Calculation**:

* + The function fibonacci(n) is called recursively.
  + The #pragma omp taskwait directive ensures that the current task waits for the completion of its child tasks before proceeding.

**2. parallel executes**

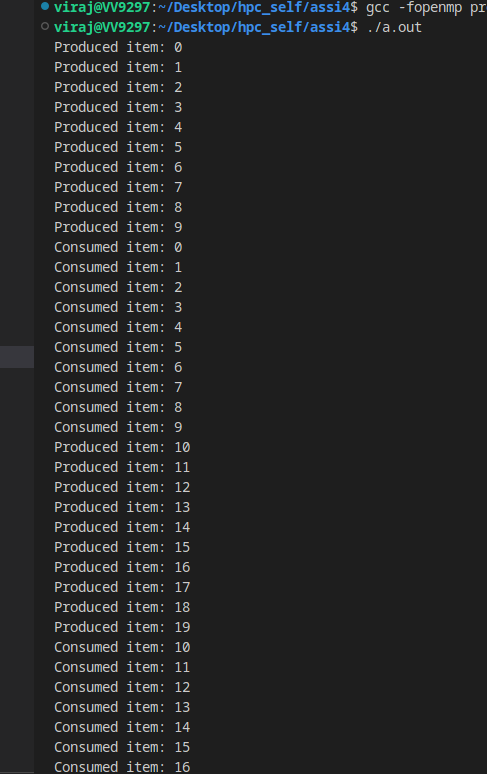
* + The #pragma omp parallel directive creates a parallel region.
  + The #pragma omp single directive ensures that only one thread executes the fibonacci(n) function call.

**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:**

This code uses OpenMP to parallelize the producer-consumer problem. The #pragma omp parallel sectionsdirective creates a parallel region with two sections, allowing the [producer](vscode-file://vscode-app/usr/share/code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)and [consumer](vscode-file://vscode-app/usr/share/code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html)functions to run concurrently. The #pragma omp sectiondirective specifies the code blocks for each section. Critical sections (#pragma omp critical) ensure mutual exclusion when accessing shared variables like [buffer](vscode-file://vscode-app/usr/share/code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [in](vscode-file://vscode-app/usr/share/code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), [out](vscode-file://vscode-app/usr/share/code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), and [count](vscode-file://vscode-app/usr/share/code/resources/app/out/vs/code/electron-sandbox/workbench/workbench.html), preventing data races. This setup allows safe concurrent execution of the producer and consumer functions.

**Github Link:**

https://github.com/Virajpatil092/hpc\_self