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

**Year:** 2023-24 **Semester:** 1

**Course:** High Performance Computing Lab

**Practical No. 4**

**Exam Seat No:2020BTECS00022**

**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:

#include <stdio.h>

#include <time.h>

#include <omp.h>

int main()

{

     long n = 0;

     printf("\nEnter number = ");

     scanf("%d", &n);

     int ans[n];

     int f1 = 1;

     int f2 = 1;

     clock\_t st = clock();

     printf("%d %d ", f1, f2);

#pragma omp parallel for schedule(guided, 4) num\_threads(8)

     for (int i = 2; i <= n; i++)

     {

#pragma omp critical

          {

               int f3 = f1 + f2;

               printf("%d ", f3);

               f1 = f2;

               f2 = f3;

          }

     }

     clock\_t et = clock();

     double elapsed\_time = (double)(et - st) / CLOCKS\_PER\_SEC;

     double elapsed\_miliseconds = elapsed\_time \* 1000;

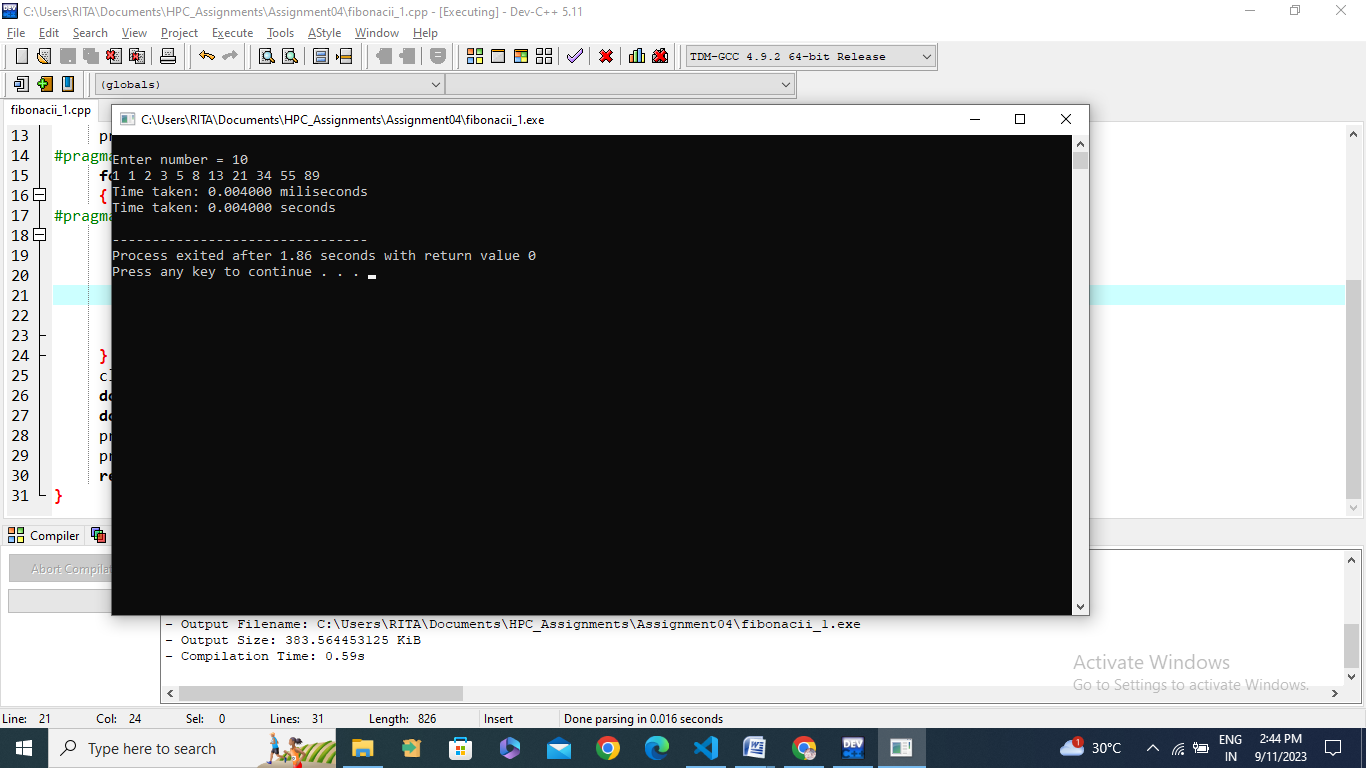
     printf("\nTime taken: %f miliseconds", elapsed\_miliseconds);

     printf("\nTime taken: %f seconds\n", elapsed\_time);

     return 0;

}

**Screenshots:**

****

**Information:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No of Threads** | **Fibonacci no** | **Sequential time** | **Parallel time** |  |
| **8** | **10** | **0.004000** | **0.004000** |  |
| **8** | **25** | **0.006000** | **0.006000** |  |
| **8** | **45** | **0.006000** | **0.006000** |  |
| **2** | **10** | **0.002000** | **0.002000** |  |
| **4** | **10** | **0.002000** | **0.003000** |  |
| **10** | **10** | **0.002000** | **0.004000** |  |

**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

#include <stdio.h>

#include <stdlib.h>

int mutex = 1;

int full = 0;

int empty = 10, x = 0;

void producer()

{

     --mutex;

     ++full;

     --empty;

     x++;

     printf("\nProducer produces item %d", x);

     ++mutex;

}

void consumer()

{

     --mutex;

     --full;

     ++empty;

     printf("\nConsumer consumes item %d", x);

     x--;

     ++mutex;

}

int main()

{

     int n;

     printf("\n1. Producer"

            "\n2. Consumer"

            "\n3. Exit");

#pragma omp critical

     while (1)

     {

          printf("\nEnter your choice:");

          scanf("%d", &n);

          switch (n)

          {

          case 1:

               if ((mutex == 1) && (empty != 0))

               {

                    producer();

               }

               else

               {

                    printf("Buffer is full!");

               }

               break;

          case 2:

               if ((mutex == 1) && (full != 0))

               {

                    consumer();

               }

               else

               {

                    printf("Buffer is empty!");

               }

               break;

          case 3:

               exit(0);

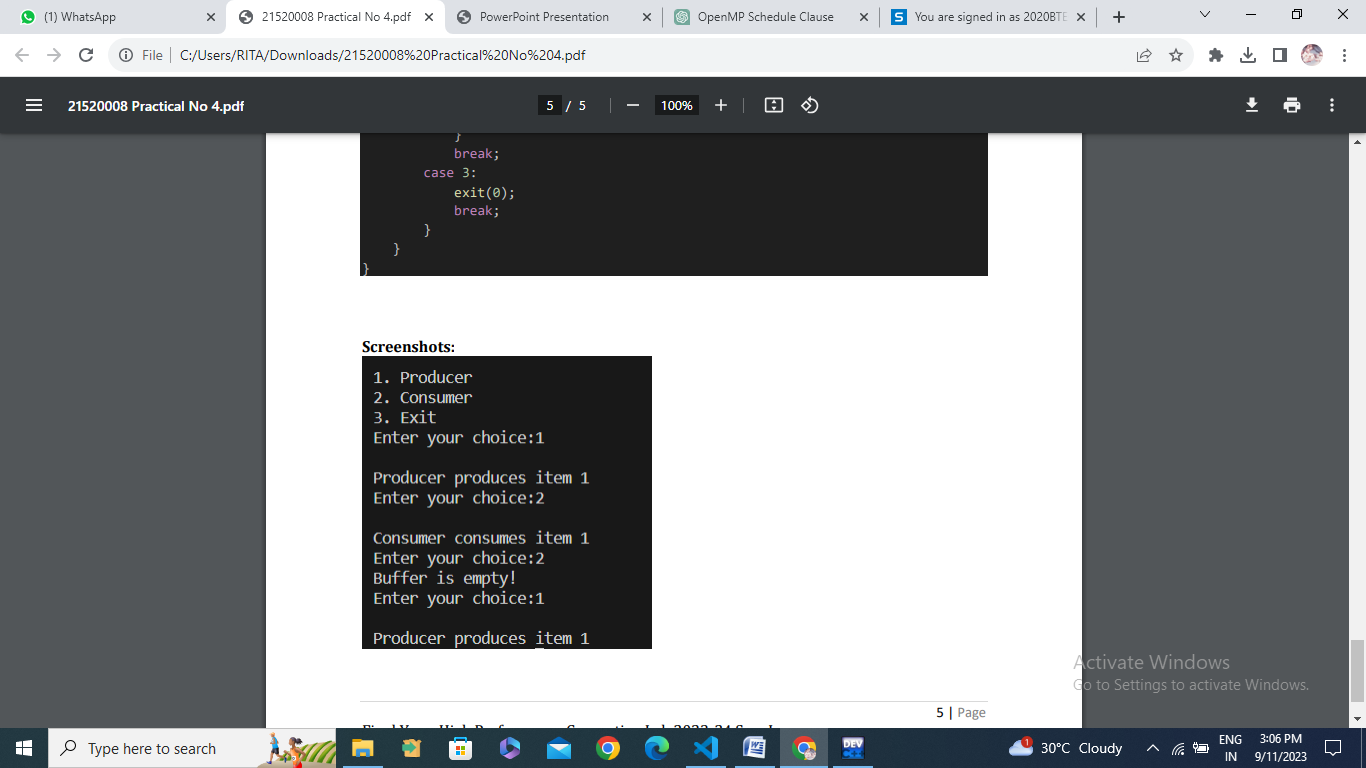
               break;

          }

     }

}

**Screenshots:**

****

**Information:**

**Github Link:** <https://github.com/GauravP07/HPC_Assignments>