Walchand College of Engineering, Sangli
Department of Computer Science and Engineering

Class: Final Year (Computer Science and Engineering)

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

Course: High Performance Computing Lab

Practical No. 4

Exam Seat No: 2020BTECS00037

Title of practical:

Study and Implementation of Synchronization

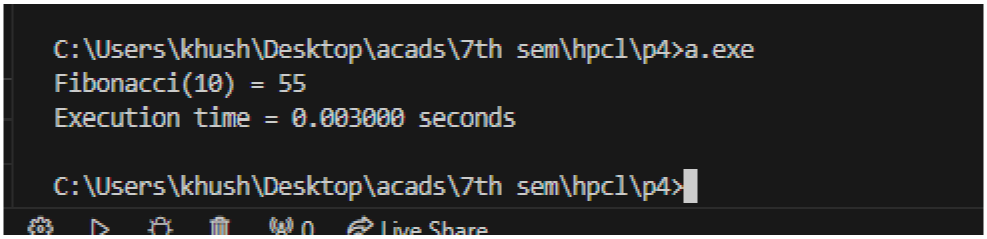
Problem Statement 1:

Fibonacci Computation:

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

```
#include <stdio.h>
#include <omp.h>

long long fib(int n) {
    if (n <= 1) {
        return n;
    } else {
        long long x, y;
        #pragma omp task shared(x)
        x = fib(n - 1);

        #pragma omp task shared(y)
        y = fib(n - 2);

        #pragma omp taskwait
        return x + y;
    }
}

int main() {
    int n = 10; // Fibonacci number to compute
    long long result;
    double start_time, end_time;

    start_time = omp_get_wtime();

    #pragma omp parallel
    #pragma omp single
    {
        result = fib(n);
    }

    end_time = omp_get_wtime();
    double execution_time = end_time - start_time;

    printf("Fibonacci(%d) = %lld\n", n, result);
    printf("Execution time = %lf seconds\n", execution_time);

    return 0;
}
```

Screenshots:



```
C:\Users\khush\Desktop\acads\7th sem\hpcl\p4>a.exe
Fibonacci(10) = 55
Execution time = 0.003000 seconds

C:\Users\khush\Desktop\acads\7th sem\hpcl\p4>
```

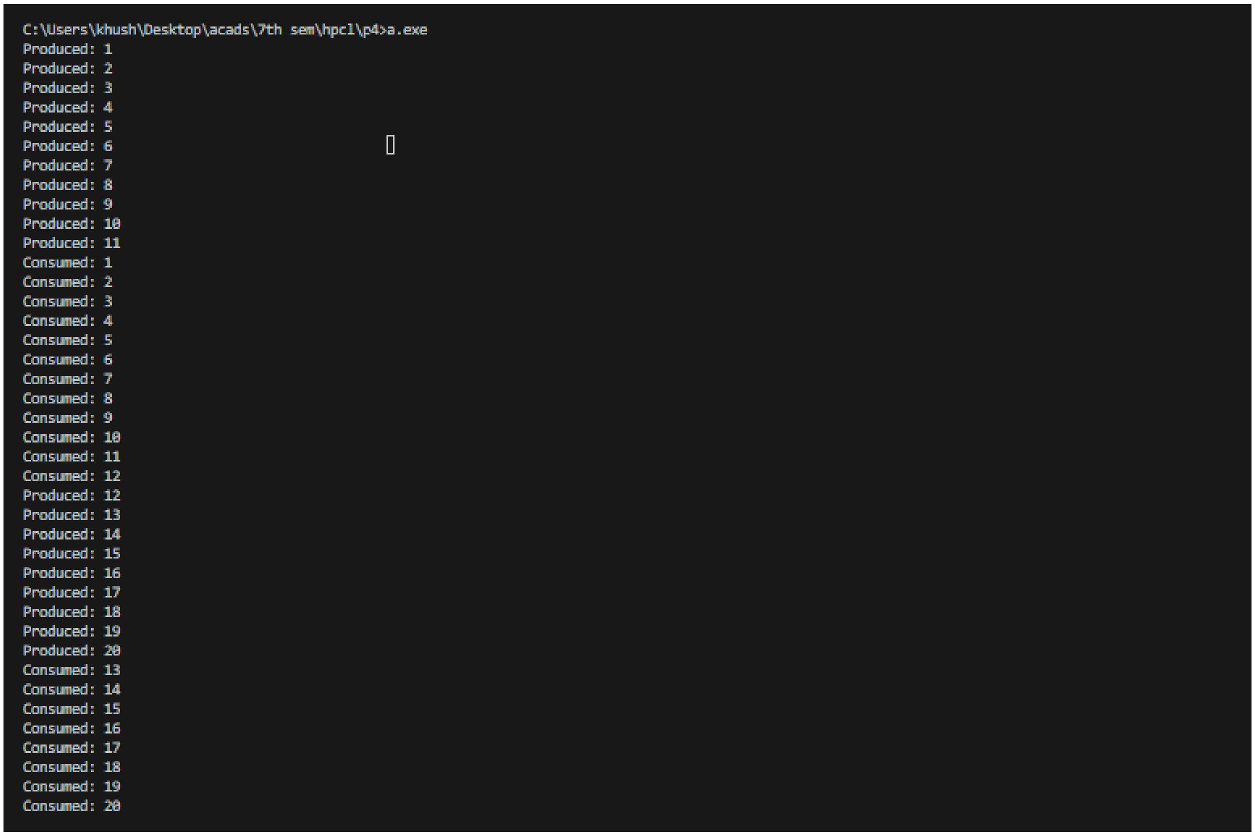
Problem Statement 2:

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>
#include <omp.h>

#define BUFFER_SIZE 10

int buffer[BUFFER_SIZE];
int count = 0; // Number of items in the buffer
int in = 0;    // Index for adding items to the buffer
int out = 0;   // Index for removing items from the buffer

void producer() {
    // Produce 20 items
    for (int i = 0; i < 20; i++) {
        while (count == BUFFER_SIZE) {
#pragma omp flush(count)
        }

        buffer[in] = i;
        in = (in + 1) % BUFFER_SIZE;

#pragma omp atomic
        count++;

        printf("Produced: %d\n", i + 1);
    }
}

void consumer() {
    // Consume 20 items
    for (int i = 0; i < 20; i++) {
        while (count == 0) {
#pragma omp flush(count)
        }

        int item = buffer[out];
        out = (out + 1) % BUFFER_SIZE;

#pragma omp atomic
        count--;

        printf("Consumed: %d\n", item + 1);
    }
}

int main() {
#pragma omp parallel sections
    {
#pragma omp section
        {
            producer();
        }

#pragma omp section
        {
            consumer();
        }
    }

    return 0;
}
```

Screenshots



```
C:\Users\khush\Desktop\acads\7th sem\hpcl\p4>a.exe
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Produced: 5
Produced: 6
Produced: 7
Produced: 8
Produced: 9
Produced: 10
Produced: 11
Consumed: 1
Consumed: 2
Consumed: 3
Consumed: 4
Consumed: 5
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Consumed: 10
Consumed: 11
Consumed: 12
Produced: 12
Produced: 13
Produced: 14
Produced: 15
Produced: 16
Produced: 17
Produced: 18
Produced: 19
Produced: 20
Consumed: 13
Consumed: 14
Consumed: 15
Consumed: 16
Consumed: 17
Consumed: 18
Consumed: 19
Consumed: 20
```