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

Class: Final Year (Computer Science and Engineering)

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

Course: High Performance Computing Lab

Practical No. 4

Exam Seat No: 21510111

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:

1.c 2.c tmp.txt

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

void fun(int n){
  int arr[n];
  arr[0] = 0;
  arr[1] = 1;

  #pragma omp for schedule(dynamic, 1)
  for(int i=2; i<n; i++){
    #pragma omp critical
    {
      arr[i] = arr[i-1] + arr[i-2];
      printf("F(%d) = %d -- computed by thread %d\n", i, arr[i], omp_get_thread_num());
    }
  }

  #pragma omp barrier

  return;
}

int main(int argc, char *argv[])
{
  fun(10);
  return EXIT_SUCCESS;
}
```

Information and output:

```
*[main][~/acad/hpc_lab/as4]$ gcc -fopenmp 1.c -o 1 && ./1
F(2) = 1 -- computed by thread 0
F(3) = 2 -- computed by thread 0
F(4) = 3 -- computed by thread 0
F(5) = 5 -- computed by thread 0
F(6) = 8 -- computed by thread 0
F(7) = 13 -- computed by thread 0
F(8) = 21 -- computed by thread 0
F(9) = 34 -- computed by thread 0
```

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:

1.c 2.c tmp.txt

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

#define BUFFERSIZE 5

int bufferSize = 5;
int numItems = 10;
int buffer[BUFFERSIZE];
int count = 0;

void producer(){
  for(int i=0; i<numItems; i++){
    int item = i;
    int produced = 0;
    while(!produced){
      #pragma omp critical
      {
        if(count < bufferSize){
          buffer[count] = item;
          printf(" + Produced item : %d\n", item);
          count++;
          produced = 1;
        }
      }
      #pragma omp flush
    }
  }
}

void consumer(){
  for(int i=0; i<numItems; i++){
    int item;
    int consumed = 0;
    while (!consumed) {
      #pragma omp critical
      {
        if(count > 0){
          item = buffer[count-1];
          printf(" - Consumed item : %d\n", item);
          count--;
          consumed = 1;
        }
      }
      #pragma omp flush
    }
  }
}
```

```c
int main(int argc, char *argv[])
{
  omp_set_num_threads(2);

  #pragma omp parallel sections
  {

    #pragma omp section
    {
      producer();
    }

    #pragma omp section
    {
      consumer();
    }

  }
  return EXIT_SUCCESS;
}
```

Information and output:

```
*[main][~/acad/hpc_lab/as4]$ gcc -fopenmp 2.c -o 2 && ./2
 + Produced item : 0
 + Produced item : 1
 + Produced item : 2
 + Produced item : 3
 + Produced item : 4
 - Consumed item : 4
 - Consumed item : 3
 - Consumed item : 2
 - Consumed item : 1
 - Consumed item : 0
 + Produced item : 5
 + Produced item : 6
 + Produced item : 7
 + Produced item : 8
 + Produced item : 9
 - Consumed item : 9
 - Consumed item : 8
 - Consumed item : 7
 - Consumed item : 6
 - Consumed item : 5
```

Github Link: https://github.com/Sidd-77/hpc-lab/tree/main/as4