WEEKS 12 & 13: Synchronization and Disk Scheduling Programs **Date: 13/11/2020**

OBJECTIVE:

PROGRAM 1: Write a C program to implement the solution to Dining-Philosophers problem.

PROGRAM 2: Write a C program to implement FCFS disk scheduling.

- **CONCEPTS ARE ALREADY COVERED IN THEORY**
- **STUDENTS ARE ADVISED TO REFER TO THE TEXT BOOK, LECTURE MATERIAL AND PROGRAMS DEMONSTRATED IN THE CLASS TO IMPLEMENT THE GIVEN PROGRAMS.**
- **STUDENTS ARE REQUIRED TO PROVIDE PROOF OF CONDUCTION (AS PER SUBMISSION PROCEDURE BELOW).**

SUBMISSION:

1. **The source code files for both PROGRAM 1 and PROGRAM 2 should be uploaded to EDMODO in WORD or ZIP FORMAT.**
2. **All the screenshots clearly showing the directory name as YOURSRN_NAME_WEEK12-13 for both the programs should be uploaded to EDMODO in a SINGLE FILE (Word or PDF format only, Do NOT zip this file).**

Students should keep these TWO deliverables (i.e. 1 & 2 above) separate and NOT zip all the files together in order to facilitate quick, timely and effective evaluation.

Contact your respective Lab faculty for any questions or clarifications needed.

DUE DATE FOR SUBMISSION: SUBMIT BOTH PROGRAM 1 and PROGRAM 2 TOGETHER ON OR BEFORE 21/11/2020 11:59 PM

PROGRAMS FOR EXECUTION AND SUBMISSION:

PROGRAM 1: Write a C program that simulates the Dining-Philosophers problem. The main program should create 5 threads, one for each philosopher and initialize their states to THINKING. Your program should print the philosopher id (number), what they are currently doing (THINKING, HUNGRY or EATING) in a sequence as shown in the sample output below.

There is a dining room containing a circular table with five chairs. At each chair is a plate, and between each plate is a single chopstick (or fork). In the middle of the table is a bowl of spaghetti. Sitting at the table, are five philosophers who spend most of their time thinking, but occasionally get hungry and eat so they can think some more.

In order to eat, a philosopher must pick up the two chopsticks to the left and right of his/her plate, then serve and eat the spaghetti from the plate. Thus, each philosopher is represented by the following pseudocode:

```
process P[i]
while true do
{
        THINK;
        PICKUP(CHOPSTICK[i], CHOPSTICK[i+1 mod 5]);
        EAT;
        PUTDOWN(CHOPSTICK[i], CHOPSTICK[i+1 mod 5])
}
```

A philosopher may THINK indefinitely. Every philosopher who EATs will eventually finish. Philosophers may PICKUP and PUTDOWN their chopsticks in either order, or non-deterministically, but these are atomic actions, and, of course, two philosophers cannot use a single/same CHOPSTICK at the same time.

EXPECTED OUTPUT: [You can use fork or chopstick variable name]

Your output can slightly vary depending upon your implementation and the manner in which you are taking input and how the threads are scheduled on your system.

Philosopher 1 is thinking

Philosopher 2 is thinking

Philosopher 3 is thinking

Philosopher 4 is thinking

Philosopher 5 is thinking

Philosopher 1 is Hungry

Philosopher 1 takes fork 5 and 1

Philosopher 1 is Eating

Philosopher 2 is Hungry

Philosopher 4 is Hungry

Philosopher 4 takes fork 3 and 4

Philosopher 4 is Eating

Philosopher 3 is Hungry

Philosopher 5 is Hungry

Philosopher 1 putting fork 5 and 1 down

Philosopher 1 is thinking

Philosopher 2 takes fork 1 and 2

Philosopher 2 is Eating

Philosopher 4 putting fork 3 and 4 down

Philosopher 4 is thinking

Philosopher 5 takes fork 4 and 5

Philosopher 5 is Eating
Philosopher 2 putting fork 1 and 2 down
.....
.....

PROGRAM 2: Write a C program to implement FCFS disk scheduling and calculate Total Seek time and Average seek time as shown in the output below.

Your output can slightly vary depending upon your implementation and the manner in which you take input values