Assignment 5

Operating System Lab (**CS341**)

Department of CSE, IIT Patna

Date:- 5-Feb-2018 **Time:-** 3 hours

Instructions:

1. All the assignments of part-I should be completed and uploaded by 5 pm. Marks will be deducted for the submissions made after 5 pm.
2. Markings will be based on the correctness and soundness of the outputs. Marks will be deducted in case of plagiarism.
3. **Proper indentation and appropriate comments are mandatory.**
4. You should zip all the required files and name the zip file as ***roll_no*.zip**, eg. **1501cs11.zip.**
5. Upload your assignment (**the zip file**) in the following link:
   PART 1: https://www.dropbox.com/request/9ptd9z9VwM2XobvD4jhn
   PART 2: https://www.dropbox.com/request/bbpmFogSK3tBHxQkVCXf

Part - I

Deadline:- 5-Feb-2018; 5:00 PM

1. Write 2 programs that will communicate **both ways** (***i.e*** each process can read and write) when run concurrently via semaphores.
2. Write a program of creating two threads (*pthread1* and *pthread2*) where each thread call a particular function. Apply a ***mutex lock (***acquiring a lock and releasing a lock***)*** on the function show that if a thread access the function using mutex lock the other thread cannot access the function. Sample output:
   a. *pthread1* is accessing function f().
   b. Lock acquired by *pthread1*
   c. *pthread2 cannot access function f() lock acquired by pthread1*
   d. Lock released by *pthread1*
   e. *pthread2* is accessing function f().

f. Lock acquired by *pthread2*

Part -II

Deadline:- 10-Feb-2018; 11:00 PM

In this assignment, you implement ***diners philosophers problem***. Here each philosopher grabs the two forks one by one – first the left fork, and then after some waiting the right fork. The parent process checks at regular intervals whether a deadlock has occurred. If so, it chooses a philosopher randomly and releases the fork (***the left one actually***) grabbed by him. Maintain a ***resource graph*** using shared memory. The parent process periodically checks for a deadlock (cycle) in the shared resource graph. Use ***semaphores*** for synchronization and mutual exclusion.

In both the programs, print suitable diagnostic messages, like the following:

```
Philosopher 3 starts thinking
Philosopher 2 starts eating
Philosopher 0 grabs fork 0 (left)
Philosopher 4 ends eating and releases forks 4 (left) and 0 (right)
Parent detects deadlock, going to initiate recovery
Parent preempts Philosopher 1
```

Also for each step, print the ***allocation matrix*** and ***request matrix*** for each process.