Assignment 7

Operating System Lab (CS342)
Department of CSE, IIT Patna

Date: 8-Mar-2022 **Deadline:** 11:59 PM

Instructions:

1. All the assignments should be completed and uploaded by 11:59 PM. Marks will be deducted for the submissions made after 11:59 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 (if necessary) are mandatory.
4. You should zip all the required files and name the zip file as roll_no.zip, eg. 1701cs11.zip.
5. Upload your assignment (the zip file) in the following link: https://www.dropbox.com/sh/7aqn6z29972qpro/AAA2rgGhy4QLE_eUtRwDRhO_a?dl=0

Questions:

1. Consider the following processes:

Reader Process:
If one reader is reading, others are also allowed to read. No reader should wait if the share is currently open for reading, even if there is a writer waiting to write.
If atleast one reader is reading, no other process can write.

Writer Process:
If a writer is writing, no other process can read it.
Once a writer is ready or gets access, it can write.
When one process is writing, other processes are not allowed to write.

Write a C program to implement the reader-writer problem for 5 readers and 2 writers.

2. Here are code samples for two threads that use binary semaphores. Give a sequence of execution and context switches in which these two threads can deadlock. Write a C program proposing a change to one or both of them such that it makes deadlock impossible.

```
semaphore *mutex, *data;

void A() {
        wait(mutex);
        wait(data);
        print(“process A”)
        signal(mutex);
        signal(data);
}

void B() {
        wait(data);
        wait(mutex);
        print(“process B”)
        signal(data);
        signal(mutex);
}
```

3. ***Late-Night Pizza*:** A group of students are studying for an exam. The students can study only while eating pizza. Each student executes the following loop:

```
while (true) {
        pick up a piece of pizza;
        study while eating the pizza
}
```

If a student finds that the pizza is gone, the student goes to sleep until another pizza arrives. The first student to discover that the group is out of pizza calls to order another pizza before going to sleep. Each pizza has *S* slices.

Write code to synchronise the student threads and the pizza delivery thread. Your solution should avoid deadlock and calls to order a new pizza (i.e., wake up the delivery thread) exactly once each time a pizza is exhausted. No piece of pizza may be consumed by more than one student. (Consider 8 students and 5 slices)