The Objective. The goal of this assignment is to make sure you have a good understanding of materials in Chapters 5 on different concurrency control algorithms. You should read relevant chapters and check the slides to answer the questions.

Questions. If you have a difficulty with any question, please contact me, TA or post a question on Piazza. Do not post answers on Piazza. If you want to check something of the form 'am I on right track with this question', post it as a PRIVATE question on Piazza. TA/I will make it public whenever appropriate.

How to submit the assignment. Submit your answers using pdf files via Handin. You can find the link on course’s website.

1. (5+5+5 pts) Suppose there is a resource that can be used by up to 2 processes at a time. The method AccessResource provides access to this resource. To ensure that only two processes can access resource at a time, the processes are not allowed to call AccessResource directly. Instead, they call the function RequestResource which in turn allows them to access the resource. The outline of RequestResource is as follows:

```java
RequestResource
{
    YOUR CODE
    AccessResource();
    YOUR CODE
}
```
Your task is to fill in the portion marked with 'YOUR CODE' so that only two processes can access the resource at a time even though more than two can request it at a time. (Note that what goes inside AccessResource is irrelevant to your assignment. Your job is to make sure that at most two of them can access it at a time).

a) Solve this problem using testset instruction. This solution may use busy waiting and does not have to guarantee fairness.

b) Solve this problem using semaphores. This solution may not use busy waiting and has to guarantee fairness.

c) Argue that solving this problem with monitors is not possible as stated. Based on the problem identified, give a solution to solve this problem using monitors.

2. (15 pts) A program consists of \( n \) threads. Each thread works in phases. Each thread begins in phase 1. It is required that all threads complete phase 1 before they can begin phase 2. Likewise, each thread must complete phase 2 before any thread can begin phase 3, and so on. So, the code for each thread has the following structure.

```c
for (i = 1; ;i++)
{
    Execute phase i
    WaitForBarrier(i, threadid);
}
```

The function WaitForBarrier ensures that if a thread finishes phase \( i \)too quickly then it is blocked. When all threads finish phase \( i \), they are resumed so that they can continue executing phase \( i+1 \).

Write the function WaitForBarrier(int, int) using semaphores. Explain the correctness of your solution.
3. (Extra credit: 15pts) Same as Q. 2, but write the code for WaitForBarrier(int, int) that has the following structure:

```cpp
WaitforBarrier(int phasenum, int threadnum) {
    semwait(special_sem)
    cout << "Thread " << threadnum << "finished phase " << phasenum << endl;
    sempost(special_sem)

    YOUR CODE
    semwait(special_sem)
    cout << "Thread " << threadnum << "continuing to phase " << phasenum+1 << endl;
    sempost(special_sem)
}
```

The purpose of special_sem is to ensure that the output is not messed up. You can assume that you have 5 threads.