# Section 6: Deadlocks and Starvation

## Question 1

List and briefly describe the four preconditions that must be present for a deadlock to form.

### Answer

Mutual Exclusion: Only one process may hold a protected resource at a time. No process may access a resource that is being held by a different process.

Hold and Wait: A process is allowed to hold a process while waiting to hold a process being held by a different process.

No Preemption: A process cannot be forced to release the resources it holds.

Circular Wait: A deadlock is a set of blocked process where each process is blocked waiting for a resource that is being held by another blocked process in the set.

## Question 2

Would a deadlock be possible in Figure 6.5(d) in the text if there was only a single instance of Rb? Why or why not?

### Answer

Deadlock would not be possible because the two or more instances of Ra is sufficient to allow P1 to proceed, complete execution, and release its instance of Rb allowing P2 to continue.

## Question 3

Answer question 6.7 from text.

Notice that ‘floating boundary between input and output buffers” means that given N buffers on disk, some or all of the buffers may be allocated to input data or output data, but i+o must be less than N.

### Answer

This system will deadlock if while Process P is processing an input buffer’s contents, Process I fills up the disk with unprocessed input buffers leaving no disk space for the output buffer that will be needed by Process P when it finished its processing (i.e. I = max). I & P will be blocked waiting for free disk space.

## Question 4

Describe the method presented in the book and slides of preventing deadlocks because of the “Hold and Wait” precondition.

How would this solution be applied to the memory allocation deadlock described in slide 6?

### Answer

The method of preventing hold and wait deadlocks is to have every process declare all of the resources needed during execution, and delaying the execution of a process until the resources are available and can be allocated to the process.

The memory deadlock problem could be avoided by 1) Requiring P1 declare its need for a total of 140K Bytes and P2 its need for a total of 150K bytes of memory 2) Allowing the operating system to delay the execution of either process until all the resources (memory) can be allocated to a process. This means delaying either P1 or P2’s execution until the other has finished.

## Question 5

What does it mean to ‘allow preemption’ of a process’s resources in order to break a deadlock?

What is the problem with implementing the preemption of the resources held by a process involved in a deadlock set? Hint: Rollback State.

### Answer

Preemption means to somehow cause a process in the deadlock set to release the resources it holds (has locked) allowing other processes in the deadlock set access to those resources and to make progress i.e. to break the deadlock.

The problem is that it is normally impossible to ‘roll back’ a process to the state it was in before it locked the contested resource. For example, the process may have modified local or static variables. The process might also have modified some shared / global resource such as a file or global data structure. These types of changes cannot be ‘undone’ and so preemption is impossible. Also know that DBMS allow the rollback of changes made to tables and so can rollback a transaction started by a process.

## Question 6

What are two issues that arise when implementing the Deadlock Prevention Strategies we discussed?

### Answer

1. Generally it is difficult to impossible for a process to declare (know) the resources it will request before starting its execution.
2. Reserving all the resources a process will use during its lifetime is a very inefficient use of those resources. For example, a resource will be allocated for the duration of the process’s execution even if the resource is only utilized for a small fraction of that time. For example, the process may need a resource near the end of its lifetime i.e. right before it exits but will have locked that resource for the duration of the process’s execution.

## Question 7

The dining philosopher’s problem solution in the text’s Figure 6.12 uses semaphores and suffers from a problem: A deadlock is possible if all philosophers decide to eat simultaneous and pick up their left forks.

What is the solution to this problem suggested in the book?

How can the solution be implemented using counting semaphores?

### Answer

The solution is to allow only N-1 philosophers to be seated at any time so at least one philosopher is guaranteed to acquire both forks.

The solution involves using a counting semaphore, initialized to N-1, that is used to limit the number of philosophers (processes) entering the critical section to four.

## Question 8

Describe the method presented in the book and slides of preventing deadlocks because of the “Circular Wait” precondition. Hint: Ordered Resource Locks

How would this method be applied to preventing the deadlock for the “transfer funds between two bank accounts” operation described in class? That is, an operation that locks two bank accounts with the goal of transferring funds from Account1 to Account2.

### Answer

The method of avoiding ‘circular wait’ involves assigning a ordering / priority to the system’s resources that determine the order in which the resources must be requested. This method prevents processes from requesting resources in an order that creates a circular dependency.

The example had the transfer operation lock the resources in the numerical or alphabetical order of the bank account’s ID. The two processes would always attempt to lock accounts in the same order making deadlock impossible.