Answers on Deadlocks

Is the following statement true or false?

Given two processes, A and B, and two resources R1 and R2, there will be a deadlock if A holds R1 and needs R2 to progress, and B holds R2 and needs R1 to progress.

A. Correct. True
B. Incorrect. False

Solution: A deadlock is a situation where there is a set of blocked processes waiting for resources such that there is no way to satisfy any of their requests.

Which other strategies would work to break the circular wait in the Dining Philosophers problem? Select all that apply.

a. Incorrect. Assign each chopstick a distinct number [0 to N] in counterclockwise order. Each philosopher "i" first grabs the right chopstick and then the left chopstick.
b. Incorrect. Each philosopher "i" grabs chopstick "i" and then chopstick "i+1 mod N"
c. Incorrect. Each philosopher "i" grabs chopstick "i" and then chopstick "i+1"
d. Correct. Assign each chopstick the numbers 1 and 2 in alternate order. Each philosopher "i" grabs the adjacent chopstick of lower number and then the resource of a higher number.

Solution:
All of these correct approaches cause a philosopher to request chopsticks in the opposite order of the others. In the incorrect solutions, each philosopher "i" would grab the same side chopstick first and then wait indefinitely for the other side chopstick, which is being held by philosopher "i+1".

Recall that if a program is sound, it means that the program is guaranteed to produce correct results at all stages without race conditions and synchronization issues. Which of the following statements is correct?

a. Incorrect. A sound program is always deadlock-free because it avoids all the race conditions.
b. Correct. A sound program is not necessarily to be deadlock-free because the processes can still wait for each other to release some resource.

Solution:
Consider the following counterexample - sound but not deadlock free. Each philosopher "i" in the Dining Philosophers' problem still chopstick "i" and then instead of immediately attempting to grab "i+1 mod N", the philosopher executes the following code:
// For philosopher i

1. GRAB(i)
2. CHECK if (i+1) mod N is available
    1. If Yes: GRAB ((i+1) mod N)
    2. If No: RELEASE (i)

A deadlock, or rather "anti-deadlock" is possible because there is a state where no philosopher eats. Starvation is possible because some philosopher "i" may never pass the check.

Is the following statement true or false?

Deadlock is possible if the mutual exclusion, hold & wait, and circular wait conditions exist, but there is resource preemption.

a. Incorrect. True
b. Correct. False

Solution:
If any condition for deadlock is removed, then deadlock cannot happen.

Is the following statement true or false?

Deadlock is possible if we run processes one by one, and complete each process before beginning the subsequent process.

a. Incorrect. True
b. Correct. False

Solution:
This means that there is never a cyclical wait so a precondition of deadlock is broken.

Given any resource allocation graph, if there is a cycle, there is a deadlock in the program it represents.

Is the reverse (that if there is a deadlock in the program, there must be a cycle in its resource allocation graph) also true?

a. Correct. Yes
b. Incorrect. No

Solution:
Yes, it is an if-and-only-if relationship.

Which of the following are components of the resource allocation graph but not the wait-for graph?

a. Correct. Nodes representing processes
b. Incorrect. Edges representing resources
c. Correct. Directed edges from a process P to a resource R if P is waiting for R
d. Correct. Directed edges from a resource R to a process P if R is allocated to P

Solution:
The wait-for graph only removes the nodes representing resources and only focuses on the waiting relationships between processes.

Which of the following is the more conservative resource allocation approach?

a. Correct. Picking one process at a time and always making sure there are sufficient resources for the process to finish before running the process.
b. Incorrect. Starting as many as processes to run in parallel to use most of the available resources.

Solution: Being conservative in resource allocating approach means sacrificing the concurrence for deadlocks avoidance.

Which of the following statements about allocation safe states is correct?

A. Correct. An allocation state is safe if there is a sequential ordering of processes such that all the processes will be able to finish executing without entering a deadlock state.
B. Incorrect. If an allocation state is a safe state, we can only use sequential execution to finish all the processes.
C. Incorrect. An allocation state is safe means it can never meet deadlock no matter how to allocate the resources.

Solution: The first option is the definition of a safe state. However, an allocation state is safe does not mean it has to be sequentially executed to avoiding deadlock. It also does not mean that you can assign execution in any order.

Consider a system that comprises of a total of 3 processes (A, B, and C) and 16 resources. Now the system has allocated 10 resources and there are 6 unallocated resources to be used. The table below lists the system's current allocation and the maximum needs for each process:

Process	Holding	Max Claims
A	2	15
B	3	10
C	5	11

What is the sequence of processes that will make the current state safe? Enter the sequence of processes as a sequence of uppercase letters in the space provided below. Do not enter any spaces, punctuation, or symbols. For example, if you think the correct sequence of processes is B, followed by A, followed by C, then enter BAC as your answer.

Correct Answer: CBA
Other Correct Answer: cba

Explanation:
An allocation state is safe if there is a sequential ordering of processes such that all the processes will be able to finish executing without entering a deadlock state.
In the current state, process A need 13 more resource, B 7 resource, and C 6 resource. There are 6 unallocated resources, therefore, we should execute process C first. After C finishes, there are 11 unallocated resources. So we should execute process B. After B finished, there are 14 unallocated resources. Finally, the process A is ready to be executed. In conclusion, the sequential ordering is C->B->A.

In the banker's algorithm, each process declares in advance a claim on the number of resources it will need. Process A consists of three steps: step A, step B, and step C. Which of the following will be the claim that process A, declares, given the following pseudocode? Assume all resources are interchangeable.
(1) Need 2 resources to do step A
(2) Need 10 resources to do step B
(3) Need 1 resource to do step C

(multiple choice question with one correct answer):

a. Correct. 10, because this is the maximum number of resources process A needs at once
b. Incorrect. 13, because this is the number of resources process A needs in total.
c. Incorrect. 2, because this is the number of resources process A needs to execute the first step.

Solution: The maximum number of resources that the process will ever request is 10. The resources used for steps A and C can also be used for step B.

Suppose given the initial matrix values as follow, is it a safe state? If it is, please write down the sequential execution of processes that will avoid deadlocks. For example, if the sequential execution is P0->P1->P2, please fill in P0P1P2. If it is not a safe state, please fill in None.

	Current A	Current B	Max A	Max B	Available A	Available B	M-C A	M-C B
P0	1	1	3	4	4	2	1	3
P1	0	2	6	4			6	3
P2	1	2	3	4			2	2

Solution: P2P0P1 or p2p0p1
Explanation: Based on M-C, P2 should be executed first since the available resource can only afford to run P2. After P2 is finished, there are 4 A and 4 B available therefore P0 is able to run. After P0 is finished, there are 7 A and 5 B available. At this time, there is enough resource for P1 to run. Therefore the sequential execution should be P2 -> P0 -> P1.

P1 has resource R1, and is requesting for R2 and R3.
P2 has resource R3, and is requesting for R4
P3 has resource R3 and R4, and is requesting for R1
P4 has resource R5 and is requesting for resource R6
P5 is requesting for resource R3

For the situation listed above, which of the following processes that are involved in the deadlock? Select all that apply
(multiple choice question with MULTIPLE correct answers)

A. Correct. P1
B. Correct. P2
C. Correct. P3
D. Incorrect. P4
E. Incorrect. P5

Explanation: P1 is waiting for P2,. P2 is waiting for P3, P3 is waiting for P1

P1 has resource R1, and is requesting for R2 and R3.
P2 has resource R3, and is requesting for R4
P3 has resource R3 and R4, and is requesting for R1
P4 has resource R5 and is requesting for resource R6
P5 is requesting for resource R3

For the situation listed above, which of the following resources that are involved in the deadlock? Select all that apply
(multiple choice question with MULTIPLE correct answers)

A. Correct. R1
B. Incorrect. R2
C. Correct. R3
D. Correct. R4
E. Incorrect. R5
F. Incorrect. R6

Explanation: P1 holds R1 which P3 needs. P2 holds R3 which P1 needs, P3 holds R4 which P2 needs. Even though P1 is requesting for R2, no one else is holding that.

11. Which of the following solutions to the deadlock in the Dining Philosophers problem always allow close to half the philosophers to eat simultaneously? Select the best answer.
(multiple choice question with ONE correct answers)

A. Incorrect. Make get chopstick[i] and get chopstick[i+1 mod N] atomic by surrounding with a global binary ("mutex") semaphore
Feedback: This will limit concurrency as it has a global mutex (lock) that all philosophers need to go through.
B. Correct. Reverse the order of get chopstick[i] and get chopstick[i+1 mod N] for every other philosopher
Feedback: Ideally, this allows half of the philosophers to be able to pick up two chopsticks and the other half of the philosophers need to wait.
C. Incorrect. After taking the left fork, the program checks if the right fork is available. If not, put down left fork, wait for some time, and retry
Feedback: This option may result in philosophers continuously grabbing and releasing locks, wasting resources, hence limiting concurrency.
D. Incorrect. Token ring. Each philosopher gets to eat once in a round robin fashion.
Feedback: This option will limit concurrency