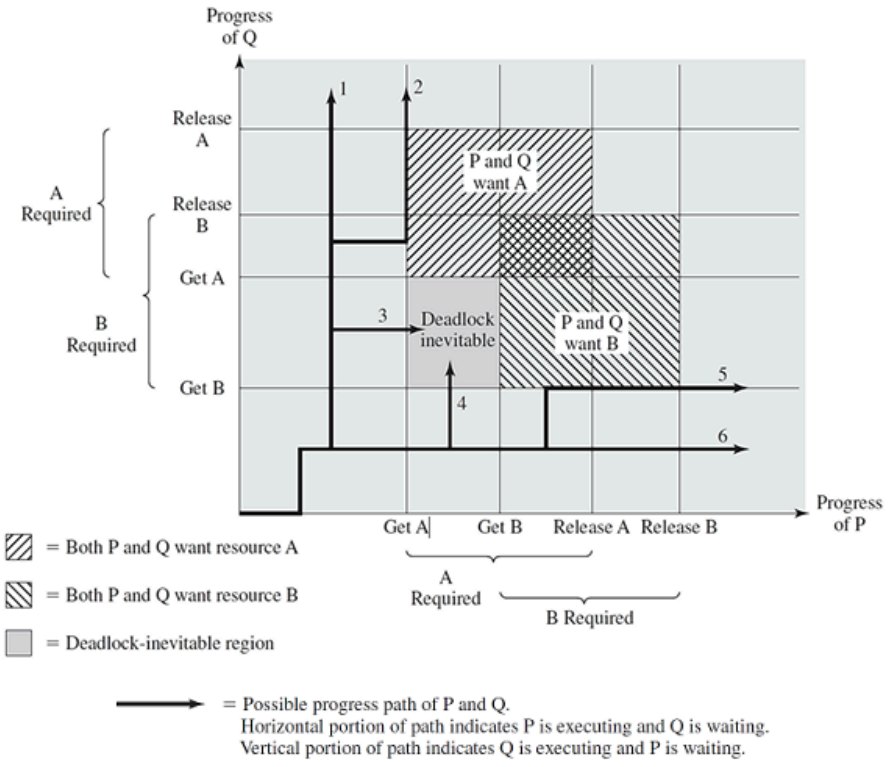
Espinase, Kristian Anthony, R.

BSCS 702

Platform Technology

04 Task Performance 1

Instructions:
Analyze the illustrations below. Then, answer the following items. Search for related literature and/or relative studies that would support your answers. Cite your references accordingly. (55 points)



1. Describe the deadlock scenario illustrated above based on your understanding

Deadlock happens when Process P and Process Q each hold one resource and wait for the other to release theirs. Neither process can move forward because they're stuck waiting. It's like two people blocking a narrow hallway, each refusing to step back. The diagram shows this clearly in the deadlock-inevitable region. Once both enter that zone, they can't continue unless something changes.

2. What do you think would happen if both Process P and Q need to get the same resource?

If both try to get the same resource at the same time, one will get it first and the other will have to wait. But if each already holds a different resource and then both ask for the same one, they might block each other. This can lead to deadlock if neither lets go. The system will freeze at that point. No process can finish its task.

3 . Which concurrency mechanism would you suggest that might prevent the deadlock situation above?

A concurrency mechanism I would suggest that might prevent deadlock is ***monitor***. It only lets one process enter at a time and keeps things organized. It also has condition checks that help decide who waits and who goes next. This avoids confusion and keeps processes from blocking each other. It's simple and safe for shared resources.

4. Define in detail the execution paths: 2 to 6

Path 2: Process P gets Resource A, then Resource B. It releases Resource B and then A.
Path 3: Process P gets Resource A. Process Q gets Resource B. Both wait for each other's resources.

Path 4: Process Q gets Resource B. Process P gets Resource A. Both wait and get stuck.
Path 5: Process Q gets Resource B, then Resource A. It releases Resource A and then B.
Path 6: Process P gets Resource B, then Resource A. It releases Resource A and then B.

5. Do Execution Paths 3 and 4 encompass the first three conditions for a deadlock to occur? Explain your answer.

Yes, they do. In both paths, each process holds one resource and waits for another. That's the hold-and-wait condition. They don't give up the resource they already have, which means no preemption. And since the resources are not shared, mutual exclusion is also present. These three conditions make deadlock possible.

6. If you are to implement deadlock prevention before the processes above reach the critical section, would it be an indirect method or a direct method?

If I were to implement a deadlock prevention before reaching the critical section, it would be an indirect method since we're stopping the first three conditions before they happen. For instance, we can make processes ask for all resources at once or give up what they hold if they can't get more. This avoids the setup that leads to deadlock. It's a way to block the problem early.

7. Which deadlock avoidance approach would you suggest for the given situation above and why?

I would suggest resource allocation denial. This means the system checks if giving a resource might cause a deadlock. If it looks risky, it says no. This keeps things safe and avoids the deadlock-inevitable region. It's better than letting processes block each other.

8. Would you agree that deadlock is relative to the number of processes and available resources in an operating system?

Yes, I agree. The more processes and fewer resources, the higher the chance of deadlock. If many processes want the same thing, they can block each other. But if there are enough resources, they won't need to wait. So it depends on how many are sharing and what's available.

9. If you are asked to reconstruct the progress diagram illustration I have attached to eliminate the critical section, which is the deadlock-inevitable region, which aspect/s or area/s would you modify? Explain how the modification eliminates the deadlock.

I would change the paths so that processes release resources before asking for new ones. This avoids hold and wait. I'd also add a rule that stops both processes from entering the deadlock region at the same time. Maybe block one until the other finishes. These changes keep the system moving and stop both from getting stuck.