OS Deadlock/Starvation Quiz

4 Conditions for deadlock	Mutual Exclusion, Hold and wait, No preemption, Circular wait
Best option for Recovery	Combine them all (Trick question :P)
Best way to Avoid Deadlock	Allow conditions 1-3 and make sure condition 4 can never happen. OS is responsible for checking to make sure deadlock can't happen before granting resource access
Circular Wait	Yielding to person to right at an intersection (rule of the road)
Circular wait	a chain of processes exists such that each process holds one or more resources needed by the next process in the chain
Common detection algorithm	Uses same matrix A and vector V. Adds new matrix Q that is how much each process is requesting. Initially consider all processes deadlocked, mark processes in A with all zeros (those can't deadlock, they aren't requesting anything) Find processes that are not requesting more than available, those are not deadlocked. Add these values to the allocated vector to keep track of what used and move to next process
Consumable Resources	Resources that are created and destroyed by one or more processes (e.g. producer consumer problem, interrupts, signals, messages, etc.)
Deadlock	permanent blocking of a set of processes that either compete for system resources or communicate with each other
Deadlock Detection	The OS periodically performs an algorithm that allows it to detect condition 4 (Circular Wait). This can be run every time a resource is requested, but tradeoff is more overhead
Direct Deadlock Prevention	Prevents Condition 4 (Circular Wait)
Disadvantage of Process initiation denial	not optimal, not efficient. Assumes all processes will make max resource claims all at once
Disadvantages of Allowing Preemption	When conflict, process releases everything it owns and re-requests it, causing inefficiency. Also will not work if any processes share a priority level as the CPU won't know who to give the desired resource to.
Disadvantages of Disallowing Hold and Wait	Inefficient, long waits occur, resources may remain allocated and unused for a long time, processes might not know what they need ahead of time
Disadvantages of Disallowing Mutual Exclusion	Simply infeasible in modern systems
Finding process that can run to completion	1. Subtract each row of [C-A] from the available vector. 2. Any process whose result row has values >= 0 can run to completion
Hold and Wait	a process can hold a resource and enter a wait state
Indirect Deadlock Prevention	Prevents conditions 1-3 (Mutual Exclusion, Hold and Wait, No Preemption).
Methods of dealing with Deadlock	Prevention, Avoidance, Detection, & Recovery
Mutual Exclusion	Only one process may use a resource at a time, other processes must wait
No Preemption	resources can not be forcibly taken from a process
Options for Recovery	Abort all deadlocked processes. Backup deadlocked processes to a previously defined checkpoint, then restart them all Start aborting deadlocked processes one by one until one is released
Process initiation denial	(requires all information known ahead of time) Puts resources into 2 matrix, available or allocated. No process is allowed more than it needs and may only take as much as the total available for each resource
Process initiation denial	Don't start a process if its demands might lead to a deadlock

24.	**Resource allocation denial**	(aka banker's algorithm) define set of system states that the OS can use to determine what allocations are safe (not result in deadlock)
25.	**Resource allocation denial**	Don't grant a resource to a process if doing so could lead to deadlock.
26.	**Reusable Resources**	Used by one process at a time and not depleted by use (e.g. processor, I/O channel, main & secondary memory, files, etc.)
27.	**Safe state**	state in which there is at least one sequence of resource allocation to process that does not result in deadlock (all processes can run to completion)
28.	**Two categories of resources that processes compete for**	Reusable and Consumable