Q1

Critical Section is the part of a program, which tries to access shared resources. That resource may be any resource in a computer like a memory location, Data structure, CPU or any IO device. It cannot be executed by more than one process at the same time; operating system faces the difficulties in allowing and disallowing the processes from entering the critical section.

The critical section problem is used to design a set of protocols, which can ensure that the Race condition among the processes will never arise. In order to synchronize the cooperative processes, our main task is to solve the critical section problem. We need to provide a solution in such a way that the following conditions can be satisfied.

Q2

A Deadlock is a situation where each of the computer process waits for a resource which is being assigned to some another process. In this situation, none of the processes is executed since the resource it needs, is held by some other process which is also waiting for some other resource to be released. Deadlocks can be avoided by avoiding at least one of the four conditions, because all this four conditions are required simultaneously to cause deadlock.

- Mutual Exclusion – Resources shared such as read-only files do not lead to deadlocks but resources, such as printers and tape drives, requires exclusive access by a single process.
- Hold and Wait – In this condition, processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.
- No Pre-emption – Pre-emption of process resource allocations can avoid the condition of deadlocks, wherever possible.
- Circular Wait – Circular wait can be avoided if we number all resources, and require that processes request resources only in strictly increasing (or decreasing) order.

Q3

The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue. The system is said to be in a safe state if there exists a sequence of other valid system states that leads to the successful completion of all processes.

- Processes request only 1 resource at a time.
- Request is granted only it results in a safe state.
- If request results in an unsafe state, the request is denied and the process continues to hold resources it has until its request can be met.
- All requests will be granted in a finite amount of time.
- Algorithm can be extended for multiple resource types.

Considering a system with five processes P_0 through P_4 and three resources types A, B, C. Resource type A has 10 instances, B has 5 instances and type C has 7 instances. Suppose at time t_0 following snapshot of the system has been taken:

Process	Allocation	Max	Available
	A B C	A B C	A B C
P_0	0 1 0	7 5 3	3 3 2
P_1	2 0 0	3 2 2	
P_2	3 0 2	9 0 2	
P_3	2 1 1	2 2 2	
P_4	0 0 2	4 3 3	

Question1. What will be the content of the Need matrix?

Need [i, j] = Max [i, j] – Allocation [i, j]

So, the content of Need Matrix is:

Process	Need		
	A	B	C
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Q4

BASIS FOR COMPARISON	PROCESS	THREAD
Basic	Program in execution.	Lightweight process or part of it.
Memory sharing	Completely isolated and do not share memory.	Shares memory with each other.
Resource consumption	More	Less
Efficiency	Less efficient as compared to the process in the context of communication.	Enhances efficiency in the context of communication.
Time required for creation	More	Less
Context switching time	Takes more time.	Consumes less time.
Uncertain termination	Results in loss of process.	A thread can be reclaimed.
Time required for termination	More	Less