TOPIC WISE QUESTIONS :-

Ex 1(Parent child communication)

Qn 1 : What are the return types of fork() ?

-1 - in case of failure in child creation
0 - to the child process
+ve value(pid of child process) - to the parent process

Qn 2 : Dual modes of OS ?

User mode and Kernel mode

Qn 3 : What is the use of pipe() ?

Inter process communication

Qn 4 : What is a system call? Give some examples.

system call is a programmatic way in which a computer program **requests a service from the kernel** of the operating system it is executed on. A system call is a way for programs to **interact with the operating system**

Eg : open, read, write, close, wait, exec, fork, exit, and kill

Qn 5 : Is fork() a system call ?

Yes

Qn 6 : If parent and child process have the same code, will the fork() in child process also create another child? If not, how?

No, the fork() in child process won't create another child.
The fork() system call creates a new process that is a copy of the calling process, including its code, data, and stack. After the fork() call, both the parent

and the child processes continue to execute **from the point of the fork() call**, and they are separate processes with their own memory spaces.

Qn 7 : Difference between fork() and vfork()

The use of **fork()** allows the execution of both processes at the **same**. On the other side, the **vfork()** system calls to **suspend the parent** process's execution **until the child process accomplishes its execution process**.

Qn 8 : What is kernel ?

The kernel is a computer program at the core of a computer's operating system and generally has complete control over everything in the system

Qn 9 : What are the OS modules in kernel ?

CPU management (Process & thread scheduling)
Memory management (Paging, Segmentation algorithms)
I/O Management
Process/Thread synchronization mechanisms (Mutexes, Spinlocks, Semaphores)
Messaging(IPC)

Qn 10 : What is kernel mode ?

Kernel mode refers to the processor mode that enables software to have **full and unrestricted access** to the system and its resources.

Qn 11 : How to switch to kernel mode ?

System calls

Qn 12 : What is PCB ? What is its use ?

A **process control block (PCB)**, also sometimes called a process descriptor, is a data structure used by computer operating systems to **store all the information about a process**.

Qn 13 : Why do we use API for system call ?

API lets the operating system manage the requests so your software is less likely to affect other software when it crashes

Qn 14 : What is daemon process ?

In computing, a daemon is a program that runs continuously as a **background process** and wakes up to handle periodic service requests, which often come from remote processes.

Qn 15 : What is zombie process? Who kills zombie process?

A process which has **finished the execution but still has entry in the process table** to report to its parent process is known as a zombie process

Qn 16 : What is orphan process?

An orphan process is a computer process whose **parent process has finished or terminated**, though it remains running itself

Qn 17 : What is booting and what is the first process which starts while booting?

Booting is the **process of starting a computer** as initiated via hardware such as a button or by a software command.

BIOS(Basic Input/Output System) will load first. It is type of firmware used to **initialize the hardware**.

Ex 2(IPC)

Qn 1 : What is the difference between shared memory and message queue ?

Shared memory - used for communication between **multiple processes** in **single processor**

Message queue - used for communication between **multiple processors**

Qn 2 : Which has more mode switches - Message queue or Shared memory ?

Message Queue

Ex 3,4(CPU scheduling)

Qn 1 : Which CPU scheduling algorithms lead to starvation ?

SJF(Shortest Job First)
Priority scheduling

Qn 2 : Examples of preemptive and non preemptive scheduling

Preemptive - FCFS, SJF, Priority
Non preemptive - SRTF, RR

Qn 3 : What is context switch ? Which CPU scheduling algorithm causes more context switches ?

A context switch is a procedure that a computer's CPU (central processing unit) follows to **change** from one task (or process) to another while ensuring that the **tasks do not conflict**
Round Robin CPU scheduling algorithm causes max number of context switches

Qn 4 : Types of schedulers

Long term scheduler - loads program into main memory
Short term scheduler - selects process to be executed
Medium term scheduler - swapping

Qn 5 : Which CPU scheduling algorithm produces maximum throughput ?

SJF(Shortest Job First) scheduling

Qn 6 : Working basis of RR(Round robin) scheduling

Round-robin scheduling allocates each task an **equal share of the CPU time**. In its simplest form, tasks are in a circular queue and when a task's allocated CPU time expires, the task is put to the end of the queue and the new task is taken from the front of the queue

Qn 7 : What is starvation ?

Starvation in operating system occurs when a process **waits for an indefinite time** to get the resource it requires

Qn 8 : What is processor affinity ?

Processor affinity or CPU pinning enables applications to bind or unbind a process or a thread to a **specific core or to a range of cores** or CPUs.

Qn 9 : What is contention scope in thread scheduling ?

The contention scope attribute is used to specify whether a thread is to compete for processing resources with other threads in the same process or with other processes in the same system

Qn 10 : What are user level and kernel level threads ?

User-Level Thread-
The User-level Threads are implemented by the user-level software. the user-level threads are basically created and implemented by the thread library which OS provides as an API for creating the managing synchronizing threads. it is faster than the kernel-level threads, it is basically represented by the program counter, stack, register, and PCB.

Example – user threads library includes POSIX threads, Mach C-Threads

Kernel-Level Thread –

So, in terms of the Operating systems basically, the threads are the unit of execution within a process. and the kernel level threads are also kinds of threads which is directly handled via kernel threads management. The Kernel-level threads are directly handled by the OS directly whereas the thread's management is done by the kernel.
In the Kernel Level Threads, Each thread is self-organizing and the kernel provides each thread with its own context with information about the thread's status, such as its name, group, and priority.

Example – The example of Kernel-level threads are Java threads, POSIX threads, etc.

Ex 5(Synchronization problems)

Qn 1 : What is a semaphore ?

Semaphores refer to the **integer** variables that are primarily used to solve the critical section problem via combining two of the atomic procedures, **wait and signal**, for the process synchronization

Qn 2 : What is busy waiting ?

Busy waiting means that a process is waiting for a condition to be satisfied in a tight **loop** without relinquishing the processor

Qn 3 : How do we achieve mutual exclusion ?

locks, semaphores, and critical sections

Qn 4 : What are race conditions ?

A race condition is an undesirable situation that occurs when a device or system attempts to perform two or more operations at the **same time**, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

Qn 5 : What are the uses of semaphore ?

Enforce mutual exclusion to prevent race conditions.
Synchronize process execution. .
Efficiently manage system resources

Qn 6 : Suggest a method to prevent busy waiting problem

Using Semaphores

Qn 7 : What are the drawbacks of Peterson's algorithm?

Busy waiting
Can run only 2 processes concurrently. So it won't function in systems with multiple CPUs

Qn 8 : Does Peterson's algorithm have busy waiting ?

Yes

Qn 9 : Why do we calculate count in reader part of program(in Reader Writer program) ?

No. of readers =1 → Writer must be blocked
No. of readers=0 → Writer must be released

Qn 10 : What is synchronization? What are its 3 conditions of a solution for synchronization?

Coordinating the execution of processes so that **no two processes access the same shared resources and data** is known as process synchronization.
The 3 conditions to be satisfied by a solution for synchronization are
Mutual exclusion
Progress
Bounded waiting

Ex 6(Deadlocks)

Qn 1 : What is the difference between deadlock prevention and deadlock avoidance ?

Deadlock prevention - **eliminating** one of the **necessary conditions** of deadlock so that only safe requests are made to OS and the possibility of deadlock is excluded before making requests

Deadlock avoidance - dynamically detecting and **avoiding situations** that could lead to deadlocks

Qn 2 : What is a safe state

A state is safe if the system can allocate all resources requested by all processes (up to their stated maximums) **without entering a deadlock state**

Qn 3 : Define deadlock

A deadlock is a situation in which two computer programs sharing the same resource are effectively **preventing** each other from accessing the resource, resulting in both programs ceasing to function.

Ex 7(Page Replacement)

Qn 1 : What is the difference between Paging and Segmentation ?

Paging - splitting of a **process** into **fixed size** divisions

Segmentation - splitting of a **process** into **variable size** divisions

Qn 2 : What is Optimal Page Replacement Algorithm ?

It minimizes the number of page faults by predicting **future accesses** and replacing the **least recently used** pages

Qn 3 : What is the difference between Internal Fragmentation and External Fragmentation ?

Internal Fragmentation - splitting **memory** into **fixed size** divisions

External Fragmentation - splitting **memory** into **variable size** divisions

Qn 4 : What is Belady’s anomaly ?

Bélády's anomaly is the phenomenon in which **increasing the number of page frames results in an increase in the number of page faults** for certain memory access patterns. This phenomenon is commonly experienced when using the **first-in first-out (FIFO) page replacement algorithm**.

Qn 5 : What is TLB ?

A **translation lookaside buffer (TLB)** is a memory cache that stores the **recent translations of virtual memory to physical memory**.

Qn 6 : What is physical address and logical address. What are the components of each? How is logical address converted into physical address ?

A **physical address** is the **actual address** in main memory where data is stored
A **logical address** is a **virtual address** generated by the CPU while a program is running
Logical address = Page offset + Page number
Physical address = Frame offset + Frame number
A logical address is converted into physical address by the **memory management unit(MMU)**
Qn 7 : Advantages of virtual memory

Balance the shortage of physical space
Inexpensive
Supports multitasking and collaboration
Avoid memory fragmentation
Enhance data security
Holds vital storage space

Qn 8 : What are the types of Address Binding ?

Compile Time Binding:Address where the program is stored is known at **compile time**.
Load Time Binding:Address is not known at compile time but known at **loading of program** i.e,before running.
Run Time Binding:Address is known at **running** of executable program

Qn 9 : What is Demand Paging?

Demand paging is a memory management technique used by operating systems to optimize memory usage. In demand paging, **only the required pages of a program are loaded into memory when needed**, rather than loading the entire program at once.

Ex 8(Disk Scheduling)

Qn 1 : What is Disk Scheduling Algorithm ?

A Process makes the I/O requests to the operating system to access the disk. Disk Scheduling Algorithm manages those requests and decides the order of the disk access given to the requests.

Qn 2 : Explain about SCAN scheduling

The scan is a disk scheduling algorithm that serves requests generated by memory management unit. It is also called an elevator algorithm. In this algorithm read and write head has to move in **one direction** and fulfill all the requests until we move to the end of the disk