Q1.

Microkernel OS structure:

This is a method to structure the operating system by removing all nonessential components from the kernel and implementing them as system and user-level programs. The result is a smaller kernel. There is little consensus regarding which services should remain in the kernel and which should be implemented in user space. Typically, microkernels provide minimal process and memory management, in addition to a communication facility. The main function of the microkernel is to provide a communication facility between the client program and the various services that are also running in user space. Communication is provided by using message passing. Ease of extending the OS.

Layered-kernel OS structure:

In layered approach, the operating system is broken into a number of layers (levels).The bottom layer (layer 0) is the hardware; the highest (layer N) is the user interface. The main advantage of the layered approach is simplicity of construction and debugging. The layers are selected so that each uses functions (operations) and services of only lower-level layers. This approach simplifies debugging and system verification. The first layer can be debugged without any concern for the rest of the system. The first layer can be debugged without any concern for the rest of the system, because, by definition, it uses only the basic hardware (which is assumed correct) to implement its functions. Once the first layer is debugged, its correct functioning can be assumed while the second layer is debugged, and so on. If an error is found during the debugging of a particular layer, the error must be on that layer, because the layers below it are already debugged. Thus, the design and implementation of the system are simplified.

Zombie Process:

A process that has terminated, but still has an entry in its parents process table as parent has not yet called wait().

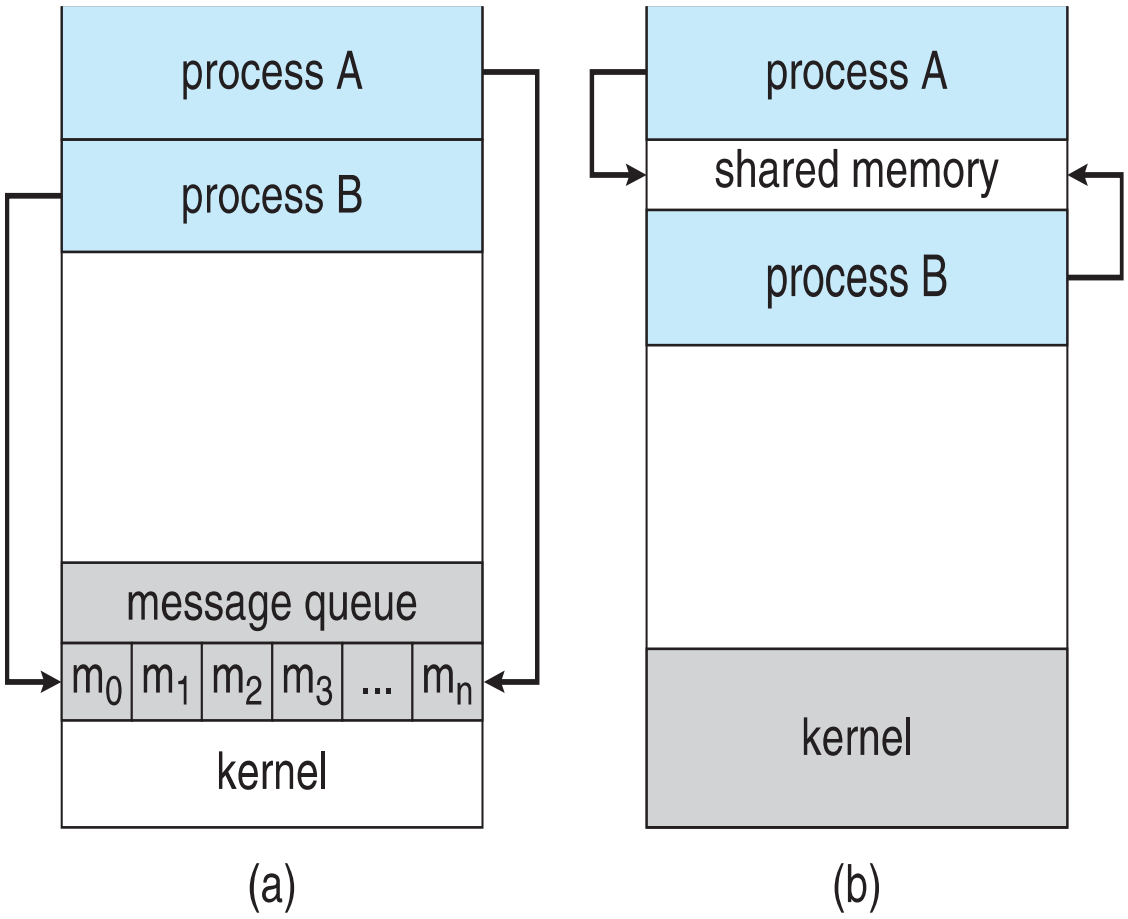
Orphan Process:

A process whose parent process terminated without waiting for its child process to terminate.

Q2: Cooperating process is the process that can affect or be affected by other processes. We need cooperative processes **for (1) Information sharing**, and (2) **Computation speedup** and **Modularity** (**using multiple child processes or threads**).

Two methods to perform IPC

1. **Shared memory (control of application)**
2. **Message passing (control of OS)**



Message Passing

Shared Memory

Q3:

FCFS: P1 at 10, P2 at 12, P3 at 15

RR: P2 at 7, P3 at 10, P1 at 15

Q4: 3.

Q5: **Preemptive scheduling:**

In this scheduling, CPU is allocated to the process for the limited amount of time and then is taken away. Which is process switches from running to ready or from waiting to ready state.

Non-Preemptive scheduling:

In this scheduling, once the CPU is allocated to a process, the process holds the CPU till it gets terminated or it reaches a waiting state.

Q6: 1. The shortest job has the highest priority.

2. FCFS gives the highest priority to the job having longest existence time.

3. None.

Q7: Creating a process requires allocating a process control block (PCB), a rather large data structure. The PCB includes a memory map, list of open files, and environment variables. Allocating and managing the memory map is typically the most time-consuming activity. Creating either a user kernel thread involves allocating a small data structure to hold a register set, stack, and priority.

Q8: 5, as the value is only changed in the parent process and not in the child process address space (or PCB).

Q9: **Turnaround time** – amount of time to execute a particular process completely

Response time – amount of time it takes from when a request was submitted until the first response is produced (first time it got the CPU), and not the final output (for time-sharing environment)

Q10: **Priority-based** scheduling algorithms and the **shortest job first** algorithm could result in starvation, since low priority processes may never execute. Shortest job first algorithm is a kind of priority scheduling algorithm where priority is the predicted next CPU burst time (larger CPU burst, lower priority).