Semaphore:

- A semaphore is a synchronization construct used in operating systems and concurrent programming to control access to shared resources.
- It acts as a signaling mechanism that allows multiple processes or threads to coordinate and communicate with one another.
- Semaphores help avoid race conditions and manage access to critical sections of code or shared resources effectively.

Key features of a semaphore:

1. Counting Mechanism: A semaphore maintains a non-negative integer value, typically known as the semaphore count. The value represents the number of available resources or permits associated with the semaphore.
2. Two Primary Operations: Semaphores support two main operations: wait (also known as P or down) and signal (also known as V or up).
3. Wait (P/Down) Operation: When a process or thread wants to access a shared resource, it must perform a wait operation on the semaphore. If the semaphore count is greater than zero, it decrements the count and continues its execution. If the count is zero, indicating no available resources, the process is blocked or put to sleep until a resource becomes available.
4. Signal (V/Up) Operation: When a process or thread releases a shared resource, it performs a signal operation on the semaphore. This increments the semaphore count, signaling that a resource has become available. If there were waiting processes, one of them will be unblocked and allowed to proceed.
5. Mutual Exclusion: By using semaphores to control access to shared resources, mutual exclusion is enforced. Only one process or thread can hold the semaphore (and thus

access the resource) at any given time, preventing conflicts and data corruption.

6. Binary Semaphore vs. Counting Semaphore: A binary semaphore is a special case of a counting semaphore, where the count can only be 0 or 1. It is typically used for simple synchronization, like mutex locks.