1. Explain three categories of concurrency.

Concurrency refers to the ability of a computer system to handle multiple tasks or processes at the same time. There are several categories of concurrency, including:

* 1. Process-based Concurrency:

This type of concurrency is achieved through the use of multiple independent processes running simultaneously on the same or different processors. Each process has its own memory space and interacts with other processes through inter-process communication (IPC) mechanisms such as pipes, sockets or shared memory.

* 1. Thread-based Concurrency:

Threads are light-weight processes that share the same memory space as their parent process. Multiple threads can be created within a single process to execute concurrently on multiple processors. Threads can communicate with each other through shared variables, and synchronization mechanisms such as locks, semaphores or barriers can be used to ensure thread safety.

* 1. Event-based Concurrency:

This approach uses an event loop to handle multiple tasks simultaneously. Tasks are triggered by events, such as user input or system notifications, and are dispatched to handlers that execute the corresponding code asynchronously. This type of concurrency is commonly used in event-driven programming frameworks and is well-suited for handling I/O-bound operations.

1. Explain the difference between heavy task using an example.

A heavy task refers to a computing task that requires significant processing power and takes a long time to complete. One example of a heavy task is video rendering or video encoding, which involves processing large amounts of data and requires dedicated hardware or software to complete in a reasonable amount of time.

For instance, consider a video production team that needs to create a high-quality animated video. The video might be several minutes or even hours long, and will need to be rendered in a specific format suitable for online streaming or distribution. This task can take a long time to complete, especially when working with high-resolution graphics or complex visual effects.

During video rendering, the computer must process all the data related to the video, from the individual frames to the sound and the effects. This requires a lot of processing power and memory, and may take hours or days depending on the length and complexity of the video.

Therefore, heavy tasks require significant computing resources and may benefit from parallel processing frameworks such as Hadoop or Spark, which allow different parts of the task to be performed simultaneously across multiple processors.

1. How do cooperation and competition synchronization work in producer-consumer problem.

In the producer-consumer problem, cooperation and competition synchronization are used to ensure that the producer and consumer processes work together efficiently and without interfering with each other.

Cooperative synchronization involves the producer and consumer cooperating with each other by synchronizing their access to a shared buffer. The producer waits until there is space in the buffer before adding new items, while the consumer waits until there are items in the buffer before consuming them. By waiting for each other in this way, the producer and consumer can communicate effectively and avoid conflicts.

Competitive synchronization, on the other hand, involves using locks or semaphores to prevent the producer and consumer from interfering with each other. In this approach, the producer locks the buffer when it is adding new items, and the consumer locks the buffer when it is consuming items. By locking the buffer in this way, the producer and consumer can work independently without interfering with each other’s work.

Overall, both cooperative and competitive synchronization can be used in the producer-consumer problem to ensure that the producer and consumer work together effectively and without conflicts, while still achieving their respective goals.

1. Using sending money in M-Pesa, explain the five states of a task.

MPESA is a popular mobile payment service in Kenya. The five states of a task in the context of sending money in MPESA can be described as follows:

Initiation: The user initiates a transaction by specifying the recipient’s phone number, the amount of money to send, and their MPESA PIN. The transaction details are validated, and the user’s account balance is checked to ensure they have sufficient funds.

Authorization: The transaction is authorized by the user by entering their MPESA PIN. The PIN is verified against the user’s registered information to ensure the authenticity of the transaction.

Processing: The transaction is processed by deducting the specified amount from the user’s account balance and transferring it to the recipient’s MPESA account. This step involves updating the account balances, generating transaction records, and ensuring that the necessary financial operations are performed securely.

Confirmation: Once the transaction is processed successfully, both the sender and recipient receive confirmation messages indicating the completion of the transaction. These messages provide details such as the transaction ID, date, time, and remaining account balance.

Completion: The task reaches the completion state when the entire process of sending money in MPESA is finished. The user can proceed with other activities or initiate new transactions if needed

1. Explain semaphore, monitor and messaging passing as design issues in concurrency. Which one fits which type of synchronization.

In concurrent programming, semaphore, monitor and messaging passing are common design issues for achieving synchronization between concurrent processes.

A semaphore is a synchronization primitive that is used for controlling access to shared resources in a multi-process or multi-threaded environment. Semaphores can be either binary (0 or 1 state) or counting (can take an arbitrary integer value) and their main property is that they can be used to indicate when a resource is available or unavailable to be used by a process. As for which type of synchronization it fits, semaphores can be used for both cooperative and competitive synchronization depending on their implementation.

A monitor is similar to a semaphore, but also includes a collection of procedures or methods that can be used to access the shared resource. Monitors typically provide a higher level of abstraction compared to semaphores, and can be used to control access to shared data structures to ensure that changes to the data are performed in a correct and consistent manner. Monitors can be seen as a simplified version of semaphores designed to handle mutual exclusion and synchronization between threads.

Message passing is a concurrency model in which processes or threads communicate by sending messages to each other. Each message contains information that specifies the action to be performed by the receiving process or thread. Message passing can be used to achieve communication and synchronization between processes or threads in a distributed environment as well as within a single system.

In terms of which type of synchronization each design issue fits, semaphores and monitors can be used for both cooperative and competitive synchronization depending on their implementation, while messaging passing is typically used for cooperative synchronization between processes or threads. The choice of which synchronization mechanism to use depends on the specific requirements of the program and the nature of the shared resources that need to be protected.