Asynchronous Programming Solutions

Asynchronous Programming

- What is the difference between starting a synchronous task and starting an asynchronous task?
  - When a task is started synchronously, we must wait for it to finish before we can perform any more operations in the current task
  - When the task is started asynchronously, we can continue performing operations in the current task, without waiting for the asynchronous task to finish

Advantages of Asynchronous Programming

- What advantages does asynchronous programming have?
  - The current task can do other work, provided it does not require the data
  - The current task only blocks when it needs the data. If the data is already available, it can continue without stopping
  - This maintains throughput and user satisfaction

Blocking and Multi-threading Programs

- Why is blocking undesirable in threaded programs?
  - Blocking reduces throughput and responsiveness of the blocked thread
  - Any threads which join with this thread will also be blocked
  - Blocking in a critical section in a critical section is particularly bad, because other threads which want to enter the critical section are blocked for longer. If we are using locks, there is a possibility of deadlock

Blocking and Non-blocking Operations

- What is meant by a blocking operation, in a multi-threaded program?
  - A blocking operation is one that stops the thread until the operation is complete
  - For example, locking a mutex, or performing an atomic operation
- Briefly explain how asynchronous programming can be used to avoid blocking operations
  - Instead of performing an operation directly, the thread puts a message on a queue
  - The operation is performed when the message is removed from the queue
  - The original thread can continue running without waiting for the operation to complete