1. Create an application that simulates downloading multiple files simultaneously.

   Use `async` and `await` to:

   a. Download three dummy files (simulated with `Task.Delay`).
   b. Log the download status to the console in real-time.

   Use `Task.WhenAll` to ensure all downloads are completed.-hitarth

2. Write an application that reads data from multiple files (use dummy text files) concurrently.-rajvi
3. Create a program that demonstrates **Asynchronous Programming** in C#. The program should utilize the **`async` and `await` keywords**, implement **Task-based asynchronous programming**, and use **Parallel processing** with `Task.Run` to execute multiple operations concurrently.-fatema

4. Create a program that performs multiple asynchronous operations concurrently. You must ensure that the program does not block while waiting for multiple tasks to complete. The twist: You need to gather and process the results of these operations **in parallel** and output them as soon as all tasks are finished.-Ishika

5.Create a program that uses `lock` to prevent multiple threads from accessing a shared resource simultaneously. The program should ensure that only one thread can modify the shared resource at any time.-Shubh

6. Write a C# program that uses `Parallel.For` to process a list of integers, performing a mathematical operation (e.g., squaring each number) in parallel. Ensure the program prints the result for each iteration after all tasks are completed.-Moxshang

7. Write a C# program with two threads. Each thread will try to acquire two locks in reverse order, causing a deadlock. Use `Thread.Sleep()` to simulate some delay between the lock acquisition. Print a message indicating when the deadlock occurs and how to resolve it.-Meghal

8. Write a program where multiple threads try to write to the same resource (e.g., increment a shared variable). Use Mutex to ensure that only one thread can access the critical section at a time. Print the value of the shared variable after all threads have completed.-Dhirav

9. Write a program that simulates a scenario where multiple threads are trying to access a limited resource (e.g., a database or API). Use `SemaphoreSlim` to limit the number of concurrent threads accessing the resource at any given time.-Utsav

10. Write a program that starts multiple asynchronous tasks and uses `Task.WhenAny()` to continue execution when the first task completes. Print the result of the first task that finishes.-Devansh

11. Write a C# program that simulates a bank account with deposit and withdrawal operations. Multiple threads should be able to perform transactions concurrently. Ensure thread safety using `lock` or `Monitor` to protect the balance from race conditions.-Vasu