1. Multithreading

Write a program to run two threads simultaneously, with each thread running a loop that prints out a message and then sleeps for one second.

2. Synchronization, Lock, Re-entrant Lock, Join

- Define a Counter class with an increment() method that increments a variable count. Use a ReentrantLock to synchronize access to the count variable.
- Also define an IncrementTask class that implements the Runnable interface. Each IncrementTask object should contain a reference to a Counter object and in its run() method, call the increment() method of the Counter object repeatedly.
- In the main() method, create three IncrementTask objects and start them in separate threads. Use the join() method to wait for all threads to finish before printing out the final count.

3. Executor-Framework

- Create an ExecutorService object using the Executors.newFixedThreadPool() method, with 3 threads.
- Submit 5 tasks to the thread pool using the executorService.submit() method.
- Each task is represented by a Task object, which implements the Runnable interface.
- The run() method of the Task object simply prints out a message indicating which task is running and on which thread it is running.

4. Future & Callable

Write a program that uses Callable to calculate the factorial of a number and returns the result using Future.

Your program should perform the following steps:

- Implement a FactorialCalculator class that implements Callable<Long>. The call() method of this class should calculate the factorial of a given number and return it.
- In your main method, create an instance of FactorialCalculator with the given number n.
- Submit the FactorialCalculator instance to an ExecutorService using the submit() method.
- Use the returned Future object to retrieve the result of the calculation.
- Print the result of the calculation.