**Контекст:** вы создали новый алгоритм майнинга криптовалюты 😊. Для реализации этого алгоритма вам нужно написать вспомогательную функцию Factorization(). Функция принимает на вход число N типа BigInteger и возвращает набор из long-чисел: это все простые множители входного числа. Если N меньше двух – исключительная ситуация. Если у N оказался простой множитель больше чем long.MaxValue – исключительная ситуация.

<https://www.wikiwand.com/ru/%D0%9F%D1%80%D0%BE%D1%81%D1%82%D0%BE%D0%B9_%D0%BC%D0%BD%D0%BE%D0%B6%D0%B8%D1%82%D0%B5%D0%BB%D1%8C>

**Постановка задачи:**

1. Написать и протестировать функцию Factorization(). Многопоточность не использовать.
2. Запустить выполнение функции Factorization() в виде отдельной задачи.
3. Превратить функцию Factorization() в асинхронную функцию FactorizationAsync() при помощи TaskCompletionSource. Попробуйте три варианта:
   1. Асинхронная работа реализуется при помощи Thread
   2. Асинхронная работа реализуется при помощи ThreadPool
   3. Асинхронности нет – функция только «притворяется» асинхронной.
4. Если у нас есть два набора простых множителей двух чисел, мы можем найти наибольший общий делитель (НОД) этих двух чисел (<https://www.wikiwand.com/ru/%D0%9D%D0%B0%D0%B8%D0%B1%D0%BE%D0%BB%D1%8C%D1%88%D0%B8%D0%B9_%D0%BE%D0%B1%D1%89%D0%B8%D0%B9_%D0%B4%D0%B5%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C>). Напишите функцию нахождения НОД (вход – два набора множителей, выход - BigInteger) и организуйте работу в виде **продолжений** задач: если задачи поиска множителей завершились успешно, запускается задача поиска НОД, затем запускается задача печати результата (этого НОД). Если хотя бы одна задача поиска множителей завершились с ошибкой, запускается задача печати этой ошибки.