**Задание 1.** Необходимо создать и протестировать в консольном приложении класс для представления кэша произвольных объектов в памяти. Для доступа к объекту в кэше используется строковый ключ. Кэш имеет ограниченную ёмкость и механизм для автоматического удаления из кэша «старых» элементов.

1. Конструктор кэша должен принимать в качестве аргумента положительное целое число. Это ёмкость кэша, то есть максимальное количество элементов в кэше.
2. Снабдите кэш методом Set(string key, object obj) для добавления объекта в кэш. Параметр key не может быть пустой строкой или null. Параметр obj не может быть равен null. Если в кэше уже есть объект с ключом key, то этот старый объект заменяется на obj (с обновлением времени последнего доступа). Если количество элементов в кэше равно его ёмкости, то перед добавлением нового объекта из кэша удаляется элемент, к которому дольше всего не обращались.
3. Снабдите кэш методом Get(string key) для получения объекта по ключу. Параметр key не может быть пустой строкой или null. Если в кэше нет объекта с ключом key, метод возвращает значение null. Если в кэше есть объект с ключом key, метод возвращает этот объект, при этом время последнего доступа к элементу кэша обновляется.
4. Создайте метод Remove(string key) для удаления из кэша объекта по ключу. Если объект нашли по ключу key и удалили, метод возвращает значение true. Если в кэше нет объекта с ключом key, метод возвращает значение false.
5. Класс кэша должен автоматически сканировать свои элементы (раз в одну секунду) и удалять те элементы, к которым не обращались более 10 секунд.

**Задание 2.** Вы создали новый алгоритм майнинга криптовалюты 😊. Для реализации этого алгоритма вам нужна функция Factorization(). Она получает в качестве аргумента число N типа BigInteger и возвращает набор чисел BigInteger – это все простые множители входного числа N (то есть, это простые числа, при перемножении которых получается число N). Если число N меньше двух – генерируется исключительная ситуация.

1. Написать и протестировать функцию Factorization(). Многопоточность не использовать.
2. Превратить функцию Factorization() в асинхронную функцию FactorizationAsync() при помощи TaskCompletionSource. Функция FactorizationAsync() должна возвращать Task<List<BigInteger>>. Пусть при этом асинхронная работа реализуется при помощи потока Thread.
3. Если у нас есть два набора простых множителей двух чисел, мы можем найти наибольший общий делитель (НОД) этих двух чисел. Вспоминаем школу (<http://spacemath.xyz/nod_i_nok/>) или ищем алгоритм в интернете. Напишите функцию GcdAsync(), которая возвращает **задачу** для нахождения НОД двух чисел типа BigInteger. GcdAsync() должна использовать в своей работе FactorizationAsync(). При тестировании функции GcdAsync() выведите в методе Main() на консоль результат (Result) возвращённой из функции задачи.

**Примечание.** Проверьте, что НОД(1234567890, 63018038201) = 1. **Ваши вычисления должны происходить за разумное время!**