Задания к работе №3 по программированию на платформе .NET.

Все задания выполняются на языке программирования C#. Версия платформы .NET - 6.0 и выше.

1. Реализуйте класс, предоставляющий функционал для токенизации текстового файла. Конструктор класса должен принимать значение типа System.Collections.Generic.HashSet<char>, хранящее в себе символы-разделители лексем текстового файла, а также строку, означающую путь к текстовому файлу в операционной системе. Метод Tokenize класса должен предоставить возможность работы с лексемами, разделёнными символами-разделителями, в цикле foreach (для класса необходимо реализовать интерфейс System.Collections.Generic.IEnumerable<string>). Обеспечьте закрытие текстового файла при разрушении объекта (реализуйте деструктор для класса) и при запросе пользователя (реализуйте интерфейс System.IDisposable). Продемонстрируйте работу реализованного класса.

2. Инфекционное отделение поликлиники имеет смотровую на *N* человек и *M* докторов, которые готовы оказать помощь заболевшим или проконсультировать здоровых людей. По правилам инфекционного отделения в смотровую может зайти здоровый человек, если в ней есть свободное место и в ней находятся только здоровые люди, и наоборот: при наличии свободных мест заболевший человек может войти в смотровую, если там только заболевшие. Как только человек вошел в смотровую он занимает свое место и ожидает доктора. Незанятый доктор выбирает пациента пришедшего раньше других и проводит прием, который длиться от 1 до *T* временных единиц. В особых случаях, доктор может попросить у другого доктора помощи, которая также может длиться от 1 до *T* временных единиц. Если все места в смотровой заняты, то пришедшие пациенты встают в очередь. При этом в очереди спустя некоторое время, при наличии заболевшего человека, заражается вся очередь. Количество пациентов и интервал их появления произволен и псевдослучаен. Реализуйте приложение, моделирующее работу инфекционного отделения с использованием средств синхронизации потоков .NET. Приложение должно вести всю историю работы инфекционного отделения: сохранять в файловую систему информацию о пришедших пациентах и их состоянии, времени работы докторов и времени оказания помощи пациентам, о новых заболевших в очереди. Продемонстрируйте работу приложения при различных значениях параметров N, M, T. Подберите параметры так, чтобы показать особые случаи, которые могут возникнуть в инфекционном отделении. Обеспечьте асинхронность операций работы с файлами.

3. Реализуйте класс универсального валидатора данных. Объект валидатора должен создаваться при помощи строителя валидатора (паттерн “строитель”), позволяющего добавить в коллекцию отдельные “атомарные” правила валидации данных (паттерн “цепочка обязанностей”) и отдать созданный валидатор, предоставляющий метод с сигнатурой void Validate(T). При попытке возврата из строителя валидатора без правил генерируйте исключительную ситуацию собственного типа. После того, как над переданными данными будут выполнены все атомарные правила, для каждого непройденного правила должна быть создана исключительная ситуация своего типа, и все такие ситуации должны быть объединены в объект типа AggregateException. Классы валидатора и строителя должны быть размещены в отдельной сборке (.dll). Продемонстрируйте работу валидатора на псевдослучайных выборках данных различных типов.

4. Реализуйте класс, представляющий собой универсальный кэш приложения. Конструктор объекта кэша принимает два параметра: время жизни записей (типа System.TimeSpan) и максимальное количество записей (типа System.Uint32). Тип кэша имеет два публичных метода Get и Save:

- при вызове метода Save переданные данные сохраняются в памяти кэша по заданному ключу (типа string); при наличии в кэше переданного ключа, должно быть сгенерировано исключение типа ArgumentException;
- вызов метода Get возвращает данные из кэша по заданному ключу (типа string); при отсутствии в кэше переданного ключа, должно быть сгенерировано исключение типа KeyNotFoundException.

Записи из кэша (вместе с ключом) удаляются, когда заканчивается их время жизни. Также, если при добавлении записи кэш имеет максимальный размер, то добавляемая запись замещает самую старую запись в кэше. Взаимодействие с кэшем возможно из разных потоков исполнения. Продемонстрируйте работу реализованного кэша.

5. Некоторое заболевание от человека к человеку передается контактным путем. Вероятность заразиться им при контакте равна p_1, $p_1 \in (0; 1]$. Заразившийся может излечиться с вероятностью p_2, $p_2 \in [0; 1]$ с получением иммунитета к заболеванию. Вспышкой болезни назовём случайное возникновение заболевания у какого-либо человека. Смоделируйте распространение заболевания (время в системе дискретно с шагом в 1 сутки), если у вас имеются данные о людях, о их знакомстве друг с другом. Формат входных данных определите самостоятельно. По результатам моделирования обеспечьте следующие возможности:

- поиск всех не заразившихся людей;
- поиск всех исцелившихся людей;
- поиск исцелившихся людей, всё окружение которых не исцелилось;
- поиск не заразившихся людей, всё окружение которых заразилось.

Окружением человека назовём всех тех, с кем этот человек непосредственно знаком.
Продемонстрируйте работу приложения при различных значениях p_1, p_2, а также различных входных данных о связях между людьми. Обеспечьте асинхронность операций работы с файлами.

6. Реализуйте приложение для умножения сверхбольших матриц (размерность матрицы намного больше 10^6). Для вашего метода умножения рассмотрите два случая: аргументы заданы в файле (текстовом или бинарном) и/или задано правило (в виде объекта делегата) по которому можно вычислить каждый элемент матрицы. Результат умножения выводите в файл. При реализации данного приложения используйте как можно больше средств работы с потоками (Thread, ThreadStart/ParametrizedThreadStart, Task/Task<T>/ValueTask/ValueTask<T>, Parallel, async/await и др.). Операции ввода/вывода также должны быть асинхронными. Реализуйте возможность просмотра входных и выходных данных. Подготовьте статистику времени работы вашего приложения для разного рода параметров. Например, для разного размера входных матриц, для разного количества потоков и т.д. Примечание. Формат вывода должен быть ясным и понятным. Программа ни при каких условиях не должна уходить в режим ожидания. Любое неясное аномальное поведение программы не допускается.