1. Реализовать класс, предоставляющий три метода расширения для интерфейса IEnumerable<T>:

- генерация всех возможных сочетаний из n (кол-во элементов перечисления) по k (с точностью до порядка, элементы могут повторяться) из элементов входного перечисления:

  Входное перечисление: [1, 2, 3]; k == 2

  Выходное перечисление: [ [1, 1], [1, 2], [1, 3], [2, 2], [2, 3], [3, 3] ]
- генерация всех возможных подмножеств (без повторений) из элементов входного перечисления:

  Входное перечисление: [1, 2]

  Выходное перечисление: [ [], [1], [2], [1, 2] ]
- генерация всех возможных перестановок (без повторений) из элементов входного перечисления:

  Входное перечисление: [1, 2, 3]

  Выходное перечисление: [ [1, 2, 3], [1, 3, 2], [2, 1, 3], [2, 3, 1], [3, 1, 2], [3, 2, 1] ]

Для каждого из методов требуется проверка элементов входного перечисления на предмет попарного неравенства по отношению эквивалентности, переданному в метод в виде реализации интерфейса IEqualityComparer<T> (если нашлись два равных по переданному отношению эквивалентности элемента, то должна быть сгенерирована исключительная ситуация типа ArgumentException). Продемонстрировать работу реализованных методов.

2. Реализовать класс логгера. Конструктор логгера принимает на вход путь к файлу, в который будут записываться логи с различными уровнями жёсткости (severity; от наименее к наиболее жёсткому: trace, debug, information, warning, error, critical; severity описан в виде enum). Метод Log позволяет залоггировать в файл переданную строку data с переданным severity в следующем формате:

[<Date> <Time>] [<severity>]: <data>

В классе реализовать необходимые интерфейсы.

3. На языке C# реализовать класс, представляющий собой универсальный (шаблонный) кэш приложения. Конструктор кэша принимает два параметра: время жизни записей (типа TimeSpan) и максимальный размер (типа int). Кэш 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. Продемонстрировать работу класса.

4. Через аргументы командной строки передаются пути к директориям. По первому пути лежат файлы формата pdf, в каждом из которых находятся вопросы экзаменационного билета (файлов произвольное количество и их имена различны; помимо pdf, могут также находиться другие файлы и директории, файлы с билетами могут находиться во вложенных директориях, вложенность произвольна), по второму - файлы формата txt (имена файлов - номера групп (номера не начинаются с символа ‘0’), в каждом файле построчно записаны ФИО студентов группы; допускается существование других файлов и директорий в заданной директории, вложенные директории обходить не нужно), по третьему - выходная директория. Необходимо выдать каждому студенту случайный билет и скопировать файл с билетом в выходную директорию, переименовав его в “<номер группы>_<Фамилия студента> <Инициалы студента>.pdf”. В случае полного совпадения имён файлов, необходимо расширить инициалы до минимального количества букв имени и/или отчества, чтобы было возможно отличить студентов. При полном совпадении ФИО и номера группы, необходимо добавить в конец имени каждого файла с билетом число от 1 до n (по числу совпавших ФИО в рамках группы). Обработайте всевозможные ошибки, которые могут произойти (по заданному пути нет файлов, файл пуст, и т. д.).

5. Описать интерфейс обобщённой (с двумя параметрами типа для ключа и значения) пирамиды (кучи), предоставляющий методы вставки данных по ключу, удаления данных по минимальному/максимальному ключу, поиска данных по ключу, слияния двух пирамид. Реализовать интерфейс в двух

классах, предоставляющих функционал двоичной пирамиды и биномиальной пирамиды соответственно, в реализациях задекорировать метод слияния при помощи определения метода operator+. Сравнение ключей задать через реализацию интерфейса IComparer<TKey>. Во время вставки данных, при наличии в пирамиде ключа, равного вставляемому, необходимо сгенерировать исключительную ситуацию типа ArgumentException. При попытке удаления данных из пустой пирамиды, необходимо сгенерировать исключительную ситуацию типа InvalidOperationException. Продемонстрировать работу реализованных классов посредством вызова методов объектов классов через ссылку на интерфейс.

6. Реализовать контейнер внедрения зависимостей со следующим функционалом:

- регистрация на интерфейс типа, реализующего этот интерфейс;
- регистрация типа на все реализованные им интерфейсы;
- регистрация маппинга от одного интерфейса к другому;
- регистрация как одиночки в контексте контейнера, либо создание объекта на каждый его запрос;
- предоставление разрешения циркумферентных зависимостей при помощи подхода setter injection;
- запрос объекта по интерфейсу, реализованному его типом;

Продемонстрировать работу контейнера. Обработать всевозможные ошибки, которые могут возникнуть в процессе регистрации метаданных в контейнере и при запросе объектов из контейнера.