1.  В классе **Task1** создайте структуру **Triangle**, представляющую собой треугольник с полем, содержащем в себе его стороны (массив целых чисел) со свойствами для чтения его сторон **A, B, C**. В конструктор треугольника передавать массив из 3-х целых чисел. Если треугольник не может быть построен, создать пустой треугольник. Напишите метод **Distinct()** для возвращения типа треугольника (равносторонний, равнобедренный или разносторонний) как строки. Напишите метод **Area()** для расчета площади треугольника: корень из р \* (р – a) \* (р – b) \* (р – c), где р – полупериметр. Переопределите метод **ToString()** для вывода информации о треугольнике в формате:  “Type = 1, subtype = 2, a = 3, b = 4, c = 5, площадь = 6”, где 1 - треугольник, 2 - тип треугольника, 3-5 - длины сторон, 6 - площадь. В классе **Task1** создайте приватное поле для массива треугольников и свойство **Triangles** для чтения этого массива. Переопределите метод **ToString()** для вывода всех элементов массива **Triangles** на консоль построчно. Напишите метод **Sorting()** для сортировки треугольников в массиве **Triangles** по возрастанию их площади (чем выше скорость сортировки, тем больше баллов за выполнение).

2. В класс **Task2** скопируйте код из класса **Task1**. Создайте в нём абстрактный класс **Figure**, представляющую геометрическую фигуру и создайте от него наследников: **Circle**, **Fourangle**, **Triangle** (уже есть).  Переопределите методы **Distinct() и Area()** для классов **Circle**, **Rectangle**, чтобы они определяли круг/эллипс и квадрат/прямоугольник и рассчитывали их площадь. В конструктор окружности передавать массив с 2 радиусами, в конструктор прямоугольника массив с двумя сторонами. В классе **Task2** создайте массив из фигур и свойство **Figures** для его чтения. Напишите метод **Sorting()** для сортировки фигур по увеличению площади. Переопределите метод **ToString()** для вывода фигур на консоль.

3. В классе **Task3** создайте класс **Uniques** с полями для входной строки и выходного массива строк. Сделайте публичные свойства для чтения этих полей **Input** и **Output**. В конструктор должен передаваться текст и сохраняться как входная строка. Найти все слова, в которых нет повторяющихся букв, и поместить в выходной массив (длина слов должна быть более 1 символа, регистр не имеет значения) и сохранить в выходное значение. Переопределите метод **ToString(),** чтобы он возвращал выходное значение построчно или пустую строку, если входной текст был неверного формата.

4. В классе **Task4** создайте абстрактрый класс **AbstractSerializer** с методами **Serialize** и **Deserialize** для записи объекта и чтения объекта по переданному пути. От него создать класс-наследник **UniquesSerializer**  и переопределить методы для работы с классом **Uniques** (или любым типом) для хранения экземпляра **Uniques** в формате JSON. В классе **Task4** создайте интерфейс **ICreator** с 4 методами: **CreateFolder** и **CreateFile**. 2 метода получают путь, где создать папку/файл и название папки/файла. 2 других метода получают путь, где создать папки/файлы и массив названий папок/файлов. Реализовать интерфейс **ICreator** в классе **UniquesSerializer**. Создайте приватное поле для хранения экземпляра Uniques и свойство для чтения этого экземпляра.