1. В классе **Task1** заполните структуру **Point**, представляющую точку в двумерном пространстве, полями для координат x и y и свойствами для их чтения **X** и **Y**. Переопределите метод **ToString()** для вывода координат точки на консоль в формате “x = 1, y = 2”, где 1 и 2 - значения полей x и y. Напишите метод **Length** для определения расстояния между точкой и переданной в метод как параметр другой точкой (округлять до 2-го знака после запятой). Напишите статический метод, который получает на вход 2 точки и возвращает в качестве строки информацию об их координатах и расстоянии между ними построчно. В конструктор передавать массив точек (и проверять, что их 2). В классе **Task1** создайте приватное поле для хранения массива точек **Point** и свойство **Points** для чтения этого поля. Переопределите метод **ToString()** для вывода всех элементов массива **Points** на консоль построчно. Напишите метод **Sorting()** для сортировки точек в массиве **Points** по возрастанию расстояния от начала координат (чем выше скорость сортировки, тем больше баллов за выполнение).

2. В классе **Task2** скопируйте код для структуры **Point** из класса **Task1**. Заполните в нем абстрактный класс **Fourangle**, представляющую четырехугольник, массивом из 4х точек **Point** и абстрактными методами **Length()** и **Area()**. В конструктор необходимо передавать массив точек Point. Если их число не равно 4-м, создавать пустой четырехугольник (все точки - нули). Переопределите метод **ToString()** для вывода фигуры на консоль в формате “a with P = b, S = c”, где a - тип фигуры, b - периметр, c - площадь. Создайте связь родитель-наследник от класса **Fourangle** для классов **Square**, **Rectangle**. Переопределите методы для вычисления периметра и площади каждой из этих фигур, используя координаты точек. В классе **Task2** создайте массив из различных фигур и свойство **Fourangles** для его чтения. Напишите метод **Sorting()** для сортировки фигур по уменьшению их площади. Переопределите метод **ToString()** для вывода фигур массива на консоль построчно.

3. В классе **Task3** заполните класс **Grep** полями для входной строки и выходной строки. Сделайте публичные свойства для чтения этих полей **Input** и **Output** соответственно. В конструктор класса **Grep** должен передаваться текст и сохраняться как входная строка. После этого нужно найти самую часто встречающуюся букву в тексте и удалить из текста все слова с этой буквой. Измененный текст сохранить как выходную строку. Переопределите метод **ToString(),** чтобы он возвращал выходную строку. В классе **Task3** создайте приватное поле для хранения экземпляра класса **Grep**, свойство **Greper**, указывающее на это поле, и переопределите метод **ToString(),** чтобы он вызывал метод **ToString()** экземпляра класса **Grep**.

4. В классе **Task4** заполните интерфейс **IDataSerializer** методами **Write** и **Read** для записи объекта и чтения объекта по переданному пути. Реализовать интерфейс в классе **DataSerializer** для работы с классом **Grep** (или обобщенным типом) для хранения экземпляра **Grep** в формате JSON. В классе **Task4** заполните интерфейс **ICreator** 2-мя методами **CreateFolder** и 2-мя методами **CreateFile**. 2 метода должны получать 2 переменные: путь, где создать папку/файл, и название папки/файла. 2 других метода получают 2 переменные: путь, где создать папки/файлы, и массив названий папок/файлов. Реализовать интерфейс **ICreator** в классе **DataSerializer**. В классе **Task4** создайте приватное поле для хранения экземпляра **Grep** и свойство для чтения этого экземпляра **Greper**.