1. В классе **Task1** создайте структуру **Number** с полем для реальной части числа (double) и свойством для его чтения **Real**. Внутри структуры реализуйте публичные методы для выполнения операций сложения, вычитания, умножения и деления чисел: **Add, Sub, Mul, Div**. Они должны принимать в себя 1 объект типа **Number** и возвращать новый объект типа **Number** после проведения операции с тем объектом, у которого был вызван метод. Переопределите метод **ToString()** для вывода числа на консоль в формате “Number = x”, где x - значение поля с реальной частью. В конструктор передавать реальную часть числа. В классе **Task1** создайте приватное поле для массива чисел **Number** и свойство **Numbers** для чтения этого массива. Переопределите метод **ToString()** для вывода всех элементов массива **Numbers** на консоль построчно. Напишите метод **Sorting()** для сортировки чисел по возрастанию (чем выше скорость сортировки, тем больше баллов за выполнение).

2. В класс **Task2** скопируйте код из класса **Task1**. Структуру **Number** переделать в класс. Создайте от него класс-наследник **ComplexNumber** с полем для мнимой части (double) и свойством для его чтения **Imagine**. Реализуйте методы для выполнения операций сложения, вычитания, умножения и деления комплексных чисел. Они должны принимать в себя 1 объект типа **ComplexNumber** и возвращать новый объект типа **ComplexNumber** после проведения операции с тем объектом, у которого был вызван метод.  Переопределите метод **ToString()** для вывода числа на консоль в формате “ Number = x ± yi”. В конструктор передавать реальную и мнимую часть числа. Переопределите метод **Sorting()** для сортировки чисел по возрастанию их модуля (re^2 + im^2)^0.5.

3. В классе **Task3** создайте класс **Searcher** 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. После этого он должен обрабатываться и помещаться в массив строк следующим образом: нужно выделить из текста все повторяющиеся слова (регистр имеет значение). Переопределите метод **ToString(),** чтобы он возвращал слова из выходного массива, разделенные пробелом.

4. В классе **Task4** создайте абстрактный класс **AbstractSerializer** с абстрактными методами **Write** и **Read** для записи объекта и чтения объекта по переданному пути. От него создать класс-наследник **SearcherSerializer** и переопределить методы для работы с классом **Searcher** (или любым типом) для хранения экземпляра **Searcher** 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 **AbstractSerializer**. Создайте приватное поле для хранения экземпляра Searcher и свойство для чтения этого экземпляра.