1. В классе **Task1** заполните структуру **Number** полями 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 хранения массива чисел **Number** и свойство **Numbers** для чтения этого массива. Напишите метод **Sorting()** для сортировки чисел по возрастанию (чем выше скорость сортировки, тем больше баллов за выполнение). Переопределите метод **ToString()** для вывода всех элементов массива **Numbers** на консоль построчно.

2. В классе **Task2** заполните класс **Number** **кодом из Task1**. Создайте для него связь родитель-наследник с классом **ComplexNumber**. В **ComplexNumber** добавьте поле для мнимой части (double) и свойстве для его чтения **Imagine**. Переопределите методы для выполнения операций сложения, вычитания, умножения и деления комплексных чисел в классе **ComplexNumber**. 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 класса передавать реальную и мнимую часть числа. В классе **Task2** создайте приватное поле для хранения массива чисел **Number** и свойство **Numbers** для чтения этого массива. Напишите метод **Sorting()** для сортировки массива чисел по возрастанию их модуля (re^2 + im^2)^0.5 (чем выше скорость сортировки, тем больше баллов за выполнение). Переопределите метод **ToString()** для вывода всех элементов массива **ComplexNumber** на консоль построчно.

3. В классе **Task3** заполните класс **Searcher** полями для входной строки и выходного массива строк. Сделайте публичные свойства для чтения этих полей **Input** и **Output** соответственно. В конструктор класса **Searcher** должен передаваться текст и сохраняться как входная строка. После этого он должен обрабатываться и помещаться в поле выходного массива: нужно выделить из текста все повторяющиеся слова (регистр имеет значение). Переопределите метод **ToString(),** чтобы он возвращал слова из выходного массива, разделенные пробелом. В классе **Task3** создайте приватное поле для хранения экземпляра класса **Searcher**, свойство **GetSearcher**, указывающее на это поле, и переопределите метод **ToString(),** чтобы он вызывал метод **ToString()** экземпляра класса **Searcher**.

4. В классе **Task4** заполните абстрактный класс **AbstractSerializer** абстрактными методами **Write** и **Read** для записи объекта и чтения объекта по переданному пути. Свяжите его с классом **SearcherSerializer** связьюродитель-наследник и переопределите методы для работы с классом **Searcher** (или обобщенным типом) для хранения экземпляра **Searcher** в формате JSON. В классе **Task4** в интерфейс **ICreator** добавьте2 метода с именем **CreateFolder** и 2 с именем **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 **AbstractSerializer**. Создайте приватное поле для хранения экземпляра **Searcher** и свойство для чтения этого экземпляра **GetSearcher**.