1.  В классе **Task1** заполните структуру **Dot**, представляющую собой точку в трехмерном пространстве (x, y, z) со свойствами для чтения её координат **X, Y, Z**. В конструктор точки нужно передавать массив целых чисел и проверять его на корректную длину. Если длина массива не равна 3, заполнять структуру нулями. Переопределите метод **ToString()** для вывода информации о точке в формате:  “x = 1, y = 2, z = 3”, , где 1, 2, 3 - значения полей x, y и z. В классе **Task1** заполните структуру **Vector**, представляющую собой вектор, составленный из двух точек **Dot** и свойствами для их чтения **A** и **B**. В конструктор передавать 2 точки: начальную и конечную. Напишите метод **Length()** для вычисления длины вектора. Переопределите метод **ToString()** для вывода информации о векторе построчно: координаты первой точки, координаты второй точки и длина вектора (“Length = дробное число“). В классе **Task1** создайте приватное поле для массива векторов и свойство **Vectors** для чтения этого массива. Переопределите метод **ToString()** для вывода всех элементов массива **Vectors** на консоль построчно. Напишите метод **Sorting()** для сортировки векторов в массиве **Vectors** по возрастанию их длины (чем выше скорость сортировки, тем больше баллов за выполнение).

2. В классe **Task2** заполните абстрактный класс **Shape**, представляющую геометрическую фигуру с полем для центра фигуры **Dot** и полем для описания радиус-вектора фигуры **Vector**, свойствами для их чтения: **Center** и **Pointer** соответственно. В конструктор класса **Shape** нужно передавать центральную точку **Dot**. Создать в классе **Shape** метод **CreateVector**, который принимает в себя точку **Dot** и заполняет поле вектора объектом **Vector**, соединяющий эту точку с центром фигуры. Создайте абстрактный метод **Volume()**. Переопределите метод **ToString()** для вывода фигуры на консоль в формате “a with V = b”, где a - тип фигуры, b - объем фигуры. Создайте связь родитель-наследник от класса **Shape** для классов **Sphere, Cube**. Переопределите метод для вычисления объёма каждой из этих фигур, используя длину вектора. В классе **Task2** создайте массив из фигур и свойство **Shapes** для его чтения. Напишите метод **Sorting()** для сортировки фигур по увеличению объема. Переопределите метод **ToString()** для вывода фигур 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 должен передаваться текст и сохраняться как входная строка. После этого нужно найти все слова-палиндромы в тексте и поместить их в выходной массив строк. Палиндром - это слово, которое читается одинаково как слева направо, так и справа налево (длина слова > 1). Переопределите метод **ToString(),** чтобы он возвращал выходное значение построчно или пустую строку, если входной текст был неверного формата.

4. В классе **Task4** заполните абстрактрый класс **SerializerInFormate** методами **Write** и **Read** для записи объекта и чтения объекта по переданному пути. Создайте связь родитель-наследник от класса **SerializerInFormate** для класса **SerializerInJson** и переопределите методы для работы с классом **Searcher** (или обобщенным типом) для сериализации экземпляра в формате JSON. В классе **Task4** заполнить интерфейс **ICreator** 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. Реализуйте интерфейс **ICreator** в классе **SerializerInJson**. Создайте приватное поле для хранения экземпляра Searcher и свойство для чтения этого экземпляра **PalindomeSearcher**.