1.  В классе **Task1** создайте структуру **Book** с полями для названия, *уникального* isbn (int, отсчет с 0), автора и года издания и свойствами для чтения этих полей **Title, ISBN, Author, Year**. В конструктор передавать все поля, кроме уникального, оно должно генерироваться автоматически. Напишите 2 статических метода для поиска книг по автору и по году с одинаковым названием **Select**. В метод передавать список книг и условие для поиска. На выход методы выводят новый список, удовлетворяющий условию. Переопределите метод **ToString()** для вывода информации о книге в формате:  “Title = 1, ISBN = 2, author = 3, year = 4”. В классе **Task1** создайте приватное поле для массива книг и свойство **Books** для чтения этого массива. Переопределите метод **ToString()** для вывода всех элементов массива **Books** на консоль построчно. Напишите метод **Sorting()** для сортировки массива **Books** по возрастанию года издания (чем выше скорость сортировки, тем больше баллов за выполнение).

2. В класс **Task2** скопируйте код из класса **Task1**. Измените структуру **Book** на абстрактный класс, добавьте ему метод для расчета стоимости **Cost(),** как 500 + количество лет, прошедших после издания. Создайте от **Book** наследников **PaperBook, ElectronicBook, AudioBook**. Для **Paper** добавьте поле твердый переплет (да/нет) и свойство **IsHardCover** для его чтения. Для **Electronic** добавьте строковое поле формат (pdf, fb2, epub) и свойство для его чтения **Format**. Для **Audio** добавьте поле студийная запись (да/нет) и свойство для его чтения **IsStudioRecord**. Переопределите метод **ToString()** для вывода информации о книге в формате:  “Type = 1, isbn = 2, spec = 3”, где 1 - тип класса, 2 - ISBN, 3 - значение дополнительного поля наследника. Переопределите метод **Cost()** с учетом дополнительных полей классов-наследников: твердая обложка + 150, для форматов множители 0.6, 0.8, 0.95, для студийной записи множитель 0.8, для не студийной - 0.6. Измените метод **Sorting()** для сортировки книг по увеличению их стоимости.

3. В классе **Task3** создайте класс **Reverter** с полями для входной строки и выходной строки. Сделайте публичные свойства для чтения этих полей **Input** и **Output**. В конструктор должен передаваться текст и сохраняться как входная строка.  После этого каждое слово текста записать задом наперед (все остальные символы должны остаться на своих местах в тексте) и поместить текст в выходную строку. Регистр букв сохранить таким, каким он был изначально. Переопределите метод **ToString(),** чтобы он возвращал выходное значение или пустую строку, если входной текст был неверного формата.

4. В классе **Task4** создайте интерфейс **ISerializer** с методами **Serialize** и **Deserialize** для записи объекта и чтения объекта по переданному пути. Создайте класс **CommonSerializer** с 4 методами для создания папок и файлов: **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. От него создать класс-наследник **JSONSerializer** и в нем реализовать интерфейс **ISerializer** для работы с классом **Reverter** (или любым типом) для хранения экземпляра **Reverter** в формате JSON. Создайте приватное поле для хранения экземпляра **Reverter** и свойство **MyReverter** для чтения этого экземпляра.