Экзаменационная контрольная работа. Вариант 1.

В отдельной библиотеке классов **«BookstoreLibrary»** объявить классы **Product**, **Book, Bookstore<T>**.

Класс **Product** описывает товар, характеризуемый стоимостью и названием, реализует интерфейс **IComparable<Product>** и содержит:
1.1) Вещественное свойство **Price** (стоимость типа **double**, должна быть положительной).
1.2) Строковое свойство **Title** (название типа **string**, не может быть пустой строкой или null).
1.3) Переопределенный метод **ToString()**, возвращающий строку формата «Price = $<Price>.» (в стоимости выводить два знака после разделителя).
1.4) Метод, переопределяющий явное приведение объекта к вещественному значению, равному стоимости текущего объекта.
1.5) Метод int **CompareTo(Product product)**, позволяющий сравнивать друг с другом объекты класса **Product** (из двух объектов бóльшим считается объект с бóльшей стоимостью).

Класс **Book** (книга) – наследник класса **Product**, дополнительно содержащий следующие члены:
2.1) Целочисленное (тип **short**) свойство – **NumberOfPages** (положительное число, количество страниц).
2.2) Целочисленное (тип **short**) свойство – **Year** (год издания книги, принимает значение в диапазоне [1990, 2020]).
2.3) Вещественное (тип double) свойство – **Rating** (рейтинг книги), принимающее значение в диапазоне [0, 5).
2.4) Метод **string GetShortInfo**() для получения краткой информации о книге, определяющейся как строка “<число страниц>.<год издания книги>.<количество уникальных символов в названии (строчное и прописное написание буквы – два различных символа)>.<рейтинг, округленный до двух знаков после запятой, без разделителя между целой и дробной частью числа>”. *Например, для книги из 500 страниц, 2001-го года издания, с названием «Harry Potter» и рейтингом 4.894 должна получиться строка* **“500.2001.9.489”**.
2.5) Переопределенный метод **ToString()**, который дополнительно к методу из базового класса возвращает значения всех свойств книги и строку из метода **GetShortInfo**().

Обобщенный класс **Bookstore<T>**, представляющий перечислимую коллекцию объектов (типизируемым параметром класса может быть только класс **Product** и производные от него). Класс **Bookstore<T>** содержит:
3.1) Закрытое поле **items** - список объектов (типа типизируемого параметра **T**) коллекции.
3.2) Метод void **Add (T item)** для добавления переданного в качестве параметра объекта к списку **items**.
3.3) Обобщенный перечислитель, возвращающий всю коллекцию объектов из **items** (используйте готовую реализацию из списка)**.**

В классах **Product** и **Book** предусмотрите проверку допустимости стоимости и дополнительных параметров (при недопустимых значениях необходимо выбрасывать исключения походящего типа с соответствующим ошибке сообщением). Разрешается расширять типы любыми членами.

В консольной программе первого проекта:
4.1) Создать переменную **books**, ссылающуюся на объект типа **Bookstore<Product>**.
4.2) Добавить к объекту **books *N*** объектов типа **Book (*N*** вводится пользователем с клавиатуры**)**, генерируя значения их свойств случайным образом (ошибки, возникающие при генерации, выводить на экран и повторять попытку генерации): **Price** – вещественное число в диапазоне [0, 20), **NumberOfPages** – целое число в диапазоне [0, 700], **Year** – целое число в диапазоне [1980, 2030), **Title** – строка случайной длины в диапазоне [3, 15], состоящая из заглавных и строчных символов латинского алфавита и пробелов, **Rating** – вещественное число в диапазоне [-2; 7).
4.3) Добавить к объекту **books** один объект типа **Product** со стоимостью равной 1 и названием “Товар1”.
4.4) Вывести содержимое **books** на экран, используя **foreach**.
4.5) С помощью **JSON**-сериализации записать содержимое переменной **books** в файл «books.json», расположенный в папке с sln-файлом решения.

В консольной программе второго проекта:
5.1) Десериализовать объект типа **Bookstore<Book>** из файла «books.json» (сформированного первым проектом) и вывести содержимое коллекции на экран. Написать три **LINQ**-запроса:
5.2) Сформировать и вывести на экран коллекцию, содержащую все книги с длиной строки краткой информации (возвращаемой GetShortInfo()) большей 14 символов, отсортированные в порядке убывания стоимости (использовать переопределенную операцию из п.1.4).
5.3) Сформировать и вывести на экран сложную коллекцию, состоящую из ***M*** других коллекций, формирующихся по принципу равенства целой части рейтинга (**Rating**) книг, где ***M*** – количество уникальных значений целых частей рейтинга (т.е. выполнить группировку по целой части рейтинга). Результаты необходимо отсортировать по возрастанию рейтинга и стоимости (использовать п.1.4). Это означает, что если всего существует 4 различных значения целых частей рейтинга, то будет сформировано 4 группы, каждая из которых содержит книги с одинаковой целой частью рейтинга, а внутри группы - сортировка по стоимости (по возрастанию).
5.4) Сформировать и вывести на экран коллекцию, содержащую только книги с максимальным годом издания и количество таких книг.

Примечание: к первостепенным подзадачам (в случае отсутствия или некорректной реализации хотя бы одной из которых максимальная оценка – 3 из 10) относятся: создание объекта (п. 4.1, 4.2), вывод необходимого через foreach (п.4.4), успешная сериализация (п.4.5) и десериализация с выводом (п.5.1), хотя бы один корректный LINQ-запрос (из п.5.2-5.4).
По оцениванию:
1 правильный LINQ – 4-5;
2 правильных LINQ – 6-7;
3 правильных LINQ – 8+.