Задача 1: Фильтрация данных

Описание: У вас есть список студентов с их оценками. Используйте LINQ, чтобы найти всех студентов, чья оценка выше 85.

```
public class Student
{
    public string Name { get; set; }
    public int Score { get; set; }
}

var students = new List<Student>
{
    new Student { Name = "Alice", Score = 90 },
    new Student { Name = "Bob", Score = 80 },
    new Student { Name = "Charlie", Score = 88 },
    new Student { Name = "David", Score = 92 }
};

// Используйте LINQ, чтобы найти всех студентов с оценкой выше 85 и выведите их имена и оценки.
```

Задача 2: Сортировка данных

Описание: У вас есть список книг. Используйте LINQ, чтобы отсортировать книги по названию в алфавитном порядке.

```
public class Book
{
    public string Title { get; set; }
}

var books = new List<Book>
{
    new Book { Title = "1984" },
    new Book { Title = "To Kill a Mockingbird" },
    new Book { Title = "The Great Gatsby" }
```

```
};
```

// Используйте LINQ, чтобы отсортировать книги по названию и вывести их в отсортированном порядке.

Задача 3: Группировка данных

Описание: У вас есть список продуктов с категориями. Используйте LINQ, чтобы сгруппировать продукты по категориям.

```
public class Product
{
    public string Name { get; set; }
    public string Category { get; set; }
}

var products = new List<Product>
{
    new Product { Name = "Apple", Category = "Fruits" },
    new Product { Name = "Carrot", Category = "Vegetables" },
    new Product { Name = "Banana", Category = "Fruits" },
    new Product { Name = "Broccoli", Category = "Vegetables" }
};
```

// Используйте LINQ, чтобы сгруппировать продукты по категориям и вывести каждую группу.

Задача 4: Объединение данных

Описание: У вас есть два списка: один с информацией о студентах, а другой — с их оценками. Используйте LINQ, чтобы объединить эти списки по идентификатору студента.

```
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
}

public class Grade
{
```

```
    public int StudentId { get; set; }
    public string Subject { get; set; }
    public char LetterGrade { get; set; }
}

var students = new List<Student>
{
    new Student { Id = 1, Name = "Alice" },
    new Student { Id = 2, Name = "Bob" }
};

var grades = new List<Grade>
{
    new Grade { StudentId = 1, Subject = "Math", LetterGrade = 'A' },
    new Grade { StudentId = 2, Subject = "Math", LetterGrade = 'B' },
    new Grade { StudentId = 1, Subject = "Science", LetterGrade = 'A' }
};

// Используйте LINQ, чтобы объединить студентов и их оценки и вывести результат.
```

Задача 5: Агрегация данных

Описание: У вас есть список заказов. Используйте LINQ, чтобы подсчитать общую сумму всех заказов.

```
public class Order
{
    public decimal Amount { get; set; }
}

var orders = new List<Order>
{
    new Order { Amount = 150.00m },
    new Order { Amount = 200.00m },
    new Order { Amount = 75.00m }
};
```

// Используйте LINQ, чтобы подсчитать общую сумму всех заказов и вывести её.