ТЗ:

Имеется класс **RefItem** представляющий из себя связный список. Реализовать **статический** метод **Revert** этого класса, который переворачивает связный список. Целостность исходного списка не принципиальна.

In: 3 -> 2 -> 4 -> 5
Out: 5 -> 4 -> 2 -> 3

```
public class RefItem<T>
{
public T Data { get; set; }
public RefItem<T> Next { get; set; }
public static RefItem<T>  Revert<T>(RefItem<T>  firstItem)
{
throw new NotImplementedException();
}
        }
```

Мат часть:

Связный Список(linked list LL) – это одна из структур классических структур данных.

Список можно сравнить с обычной очередью в магазин. Если каждый стоящий в очереди запомнит, кто за ним стоит, после чего все в беспорядке рассядутся на лавочке, получится односторонне связанный список; если он запомнит еще и впереди стоящего, будет двусторонне связанный список.

Когда последний в списке указывает на первого, то получится кольцевой список. Самый простой пример из жизни, это список песен в медиаплеере. Когда заканчивается последняя песня, то начинается воспроизведение с начала.

В LL элементы линейно упорядочены, но порядок определяется не номерами, как в массиве, а указателями на соседние элементы, входящими в состав элементов списка. Такая организация позволяет располагать данные в любой области памяти.

Односвязный список хранит указатель(ключ) предназначенный для идентификации следующего элемента. В двусвязном находится два указателя-на предыдущий и на следующий элемент списка. Остальные поля интерпретируются как дополнительные данные(satellite data), хранящиеся вместе с указателями.

Реализации структуры в форме списков особо полезно в тех случаях, когда данные в памяти необходимо запомнить сразу же, и заранее неизвестно сколько данных может прийти.

Как правило реализуются такие методы, как:

1. Поиск();
2. Вставка();
3. Сортировка();
4. Проверка элемент на принадлежность классу
5. Удаление

Анализ задания:

В ТЗ задании указано, что имеется класс, представляющий LL. Из предложенном ниже в ТЗ исходного кода, видно, что класс описан на уровне прототипа.

Определены методы Data, Next, и Revert.

Метод Data – представляет данные

```
public T Data { get; set; }
```

Метод Next представляет ссылку на следующий элемент в списке

```
public RefItem<T> Next { get; set; }
```

Метод Revert – представляет реверс очередности от конца к началу(разворот списка).

```
public static RefItem<T> Revert<T>(RefItem<T> firstItem)
                {
throw new NotImplementedException();
                }
```

В коде метода **Revert** прописано исключение **NotImplementedException(),** выбрасываемое когда запрошенный метод или операция не реализованы.

Определены входные данные RefItem<T> **firstItem** – указатель на первый элемент списка с типом T.

В теле Next и Data определены ключевые слова get и set на уровне автоматически реализуемого свойства. (когда методы доступа get и set свойства не выполняют никаких иных операций, кроме задания или извлечения значения в закрытом поле, можно использовать поддержку автоматически реализуемых свойств в компиляторе C#.)

При реализации класса использован универсальный тип T (generic), обобщенный тип. Это означает, что на входе можно использовать любой из указанных типов или любой тип, являющийся более производным.

Отдельно указано, что метод **Revert** должен быть **статическим**.

Документация Microsoft описывается следующее:
Статический член вызывается для класса даже в том случае, если не создан экземпляр класса. Доступ к статическому члены всегда выполняется по имени класса, а не экземпляра. Существует только одна копия статического члена, независимо от того, сколько создано экземпляров класса. Статические методы и свойства не могут обращаться к нестатическим полям и событиям в их содержащем типе, и они не могут обращаться к переменной экземпляра объекта, если он не передается явно в параметре метода.

Переменные, объявляемые как static, по существу, являются глобальными.
Пример вызова:

```
Automobile.Drive();
int i = Automobile.NumberOfWheels;
```

Вызов статического метода генерирует инструкцию вызова в промежуточном языке Microsoft (MSIL), в то время как вызов метода экземпляра генерирует инструкцию callvirt, которая также проверяет наличие ссылок на пустые объекты. Однако в большинстве случаев разница в производительности двух видов вызовов несущественна.
Статическими стоит объявлять только методы независимые от инициализации полей данных класса. Например, такими методами могут быть, методы выводящие какую-либо статичную информацию и т.д.

Алгоритм.

Разворот связного списка наиболее оптимально, с точки зрения использования памяти выполнять через цикл а не через рекурсию.