ДЕНЬ 4А. Спецификации

4А.1. Расширить функционал заданного класса, используя наследование

Вам дан код класса, выполняющий преобразование текста

```
class TextTransformer
{
    protected string[] lines;
    public TextTransformer(string text)
    {
        lines = text.Split('\n');
    }

    virtual public string TransformWord(string word, int lineIndex, int wordIndex,
                                        int wordCount)
    {
        return System.Text.RegularExpressions.Regex.Replace(word, "[а-я]", "*");
    }

    virtual public string TransformLine(string line, int lineIndex)
    {
        string[] words = line.Split(' ');
        for (int i = 0; i < words.Length; i++)
        {
            words[i] = TransformWord(words[i], lineIndex, i, words.Length);
        }
        return string.Join(" ", words);
    }

    public string GetResult()
    {
        string[] result = new string[lines.Length];
        for (int i = 0; i < lines.Length; i++)
        {
            result[i] = TransformLine(lines[i], i);
        }
        return string.Join("\n", result);
    }
}
```

Ваша задача – поменять способ трансформации текста, создав класс, который наследуется от *TextTransformer*. В созданном подклассе для изменения поведения Вы должны переопределить методы *TransformWord* и (или) *TransformLine*. Все дополнительно созданные члены класса должны иметь спецификатор доступа private. Метод *GetResult* должен остаться без изменения.

Изменять класс *TextTransformer* недопустимо

В каждом варианте задания помимо использования нового преобразования для некоторых частей текста необходимо сохранить исходное (см. варианты).

Для демонстрации работы Вам необходимо разработать программу (консольную или оконную, на Ваш выбор) со следующей логикой:

1. пользовать вводит многострочный текст
2. создается экземпляр вашего подкласса и в конструктор передается исходный текст
3. с помощью вызова метода *GetResult* у экземпляра получается преобразованный текст
4. преобразованный текст выводится на экран

Пример

Новое преобразование: удалить каждое второе слово.

Исходное преобразование: каждая вторая строка

```
class DerivedTextTransformer: TextTransformer
{
    public DerivedTextTransformer(string text) : base(text)
    {
    }

    override public string TransformWord(string word, int lineIndex, int wordIndex,
                                         int wordCount)
    {
        if (lineIndex % 2 == 1)
        {
            return base.TransformWord(word, lineIndex, wordIndex, wordCount);
        }
        return wordIndex % 2 == 1 ? "" : word;
    }
}
```

Исходный текст:

```
В томленьях грусти безнадежной
В тревогах ШУМНОЙ суеты,
Звучал мне долго голос нежный
И снились МИЛЫЕ ЧЕРТЫ.
```

Преобразованный текст:

```
В  грусти
В ******** ШУМНОЙ *****
Звучал  долго  нежный
И ******* МИЛЫЕ ЧЕРТЫ.
```

Варианты

№	Новое преобразование	Исходное преобразование для
1	Удаление слов с цифрами	Каждое слово, которое совпадает с предыдущим словом в строке
2	Удалить знаки препинания	Каждое слово, заканчивающееся на согласную
3	Удалить все гласные	Каждое слово с дефисом
4	Перемешать в случайном порядке все слова в строке	Каждая строка, в которой присутствуют слова короче 5 символов
5	Поменять местами слова в каждой паре слов: аб вг де жз -> вг аб жз де	Каждая строка длиннее 4 слов
6	Удалить все согласные	Каждое слово длиннее 5 символов
7	Перевести в верхний регистр каждую вторую букву слов: абв гдеж -> аБв гДеЖ	Каждая строка короче 5 слов
8	Перевернуть строки (абг деж -> жед гба)	Каждая строка, начинающаяся с большой буквы
9	Удалить первое и последнее слова в строках	Каждая строка, в которой присутствуют слова длиннее 6 символов

10	Удалить все слова без гласных букв	Каждое слово, заканчивающееся на гласную
11	Удалить все слова, в которых меньше 5 согласных	Каждое слово, содержащее цифры
12	Перевести все буквы первого слова каждой строке в верхний регистр	Каждая строка, которая длиннее по количеству слов, чем предыдущая
13	Перевернуть каждое слово в строке (абг деж -> гба жед)	Каждое слово, начинающееся с большой буквы
14	Удалить все слова размером меньше, чем из 5 букв	Каждая строка, содержащая два одинаковых слова подряд
15	Удалить все слова размером больше, чем из 5 букв	Каждая строка длиннее 3 слов
16	Перевести в верхний регистр все согласные	Каждое слово длиннее 10 символов
17	Удалить все слова, в которых больше 5 согласных	Каждая строка короче 4 слов
18	Перевести первую букву каждого слова в верхний регистр	Каждое слово короче 8 символов
19	Удалить все слова без согласных букв	Каждая строка, заканчивающаяся знаком препинания
20	Перевести в верхний регистр все гласные	Каждое слово короче 5 символов

Примечание: под словами в данном задании понимаются подстроки, состоящие из любого количества любых символов (кроме пробела), разделенные одним пробелом

4А.2. Разработать модульный тест для функции проверки правильности ввода данных, возвращающей код ошибки

Необходимо разработать функцию проверки правильности ввода данных для заданной записи и модульный тест, ее проверяющий. Функция проверки правильности ввода должна принимать на вход объект записи и возвращать, либо 0, если ошибок нет, либо номер поля с ошибкой (начиная с 1)

Решение должно состоять из двух проектов:

- *Библиотека классов (.NET Framework)*, в которой должен быть класс с функцией проверки правильности ввода данных и класс, представляющий собой запись
- *Проект модульного теста (.NET Framework),* в котором должен быть реализован модульный тест.

Модульный тест должен проверить, как положительный исход функции, так и все варианты отрицательного исхода.

Варианты

№	Поля записи и их проверка		
	Поле 1	**Поле 2**	**Поле 3**
1	Фамилия (строка): не меньше 3 букв	Должность (строка): только русские буквы	Оклад в руб (целое число): больше 0
2	Название цеха (строка): только буквы и цифры	План выпуска деталей (целое число): не больше 1000	Фактический выпуск деталей (целое число): не меньше 0

3	Город (строка): не больше 255 символов	Улица (строка): только русские буквы, знак дефиса, пробела и цифры	Номер дома (число): больше 0
4	Фамилия (строка): не меньше 3 букв	Рост (вещественное число): не больше 300	Вес (вещественное число): больше 30
5	Адрес отправления (строка): не меньше 30 символов	Адрес доставки (строка): не равен адресу отправления	Вес (вещественное число): не больше 90
6	Автомобильный номер (строка): только цифры и буквы А, В, Е, К, М, Н, О, Р, С, Т, У, Х	Год выпуска (целое число): от 1980 до текущего года включительно	Пробег в км (целое число): больше или равно 0
7	Название (строка): не больше 255 символов	Число сезонов (целое число): больше или равно одному	Год выпуска первого сезона (целое число): от 2000 до текущего года включительно
8	Фамилия (строка): не меньше 3 букв	Номер группы (строка): только цифры, знак дефиса и русские буквы	Номер в группе (целое число): больше 0
9	Фамилия (строка): не меньше 3 букв	Год поступления (целое число): не меньше 1952 и не больше текущего года	Средний балл (вещественное число): от 0 до 5 включительно
10	Фамилия (строка): не меньше 3 букв	Оценка за теорию (целое число): не меньше 3	Оценка за практику (целое число): не больше 5
11	Номер заказа (строка): только латинские буквы и цифры	Описание (строка): не меньше 200 символов	Сумма заказа (целое число): не меньше 0
12	Дисциплина (строка): только русские буквы, знаки дефиса и пробела	номер курса (целое число): от 0 до 6 включительно	количество часов (целое число): от 0 до 9999 включительно
13	Номер телефона (строка): только цифры, скобки, знаки тире, плюса и пробела	Имя оператора (строка): не меньше 3 символов	баланс в копейках (целое число): не меньше -9999999
14	Название товара (строка): не меньше 10 символов	Количество на складе (целое число): от 0 до 999 включительно	Количество зарезервированных (целое число): не меньше 0
15	Название материала (строка): не меньше 10 символов	Объем (вещественное число): не больше 1000	Вес (вещественное число): не меньше 0
16	Тема письма (строка): не больше 1024 символов	Адресат (строка): только латинские буквы, знак «@», точка, тире и цифры	Число слов (целое число): не меньше 1
17	Адрес сайта (строка): только латинские буквы, точка, тире и цифры	Число посетителей (целое число): не меньше числа уникальных посетителей	Число уникальных посетителей (целое число): не меньше 0
18	Производитель (строка): не меньше 10 символов	Объем выпуска (вещественное число): не меньше 0	Средняя цена (вещественное число): от 1 до 99999 включительно
19	Компания (строка), не меньше 10 символов	Сумма поступлений в млн. руб. (вещественное число): не меньше 0 и не больше 1000000	Сумма списаний в млн. руб. (вещественное число): не меньше 0
20	Фамилия (строка): не меньше 3 букв	Число ролей (целое число): не меньше 0 и не больше 9999	Гонорар в млн. руб. (вещественное число): не меньше 0

4А.3. Создать метод, проходящий заданные модульные тесты

Необходимо реализовать метод *Calculate* для обработки массива целых чисел, который проходит заданные модульные тесты. Объявление метода должно выглядеть следующим образом:

```
public class MyTask
{
    static public int Calculate(int[] numbers)
    {
        // Тело метода
    }
}
```

Пример

Заданные тесты:

```
[TestClass]
public class UnitTest
{
    [TestMethod]
    public void CheckZero()
    {
        int[] input = new int[0];
        Assert.AreEqual(0, MyTask.Calculate(input);
    }
    [TestMethod]
    public void CheckSum()
    {
        int[] input = new int[] { 4, 8, 15, 16, 23, 42};
        Assert.AreEqual(108, MyTask.Calculate(input));
    }
}
```

Решение:

```
public class MyTask
{
    static public int Calculate(int[] numbers)
    {
        int sum = 0;
        foreach (int n in numbers) sum += n;
        return sum;
    }
}
```

Варианты

№	Тесты
1	[TestMethod] public void CheckMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4}; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(6, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(25, MyTask.Calculate(input3)); } [TestMethod]

	public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
2	[TestMethod] public void CheckMaxIndex() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(2, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(3, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(1, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMaxIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
3	[TestMethod] public void CheckAbsMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(7, MyTask.Calculate(input2));

	`` int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckException() { int[] input = new int[0]; Assert.ThrowsException<Exception>(() => MyTask.Calculate(input)); } ```
4	``` [TestMethod] public void CheckMin() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(-7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(-32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMin() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); } ```
5	``` [TestMethod] public void CheckMinIndex() { ```

```
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(3, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(0, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(3, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMinIndex()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(-5, 7);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10) };
        input[p] = v;
        Assert.AreEqual(p, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckException()
{
    int[] input = new int[0];
    Assert.ThrowsException<Exception>(() => MyTask.Calculate(input));
}
```

6

```
[TestMethod]
public void CheckMinAbs()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(1, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(2, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(4, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMin()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(-5, 7);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckMinusOne()
{
```

	`int[] input = new int[0];` `Assert.AreEqual(-1, MyTask.Calculate(input));` `}`
7	`[TestMethod]` `public void CheckNumberPositive()` `{` `int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };` `Assert.AreEqual(6, MyTask.Calculate(input1));` `int[] input2 = new int[] { -7, 5, 3, 6, 2 };` `Assert.AreEqual(4, MyTask.Calculate(input2));` `int[] input3 = new int[] { 10, 25, -4, -32, 19 };` `Assert.AreEqual(3, MyTask.Calculate(input3));` `}` `[TestMethod]` `public void CheckRandNumberPositive()` `{` `Random r = new Random();` `for (int i = 0; i < 100; i++)` `{` `int v = r.Next(-5, 7);` `int n = r.Next(5);` `int s = r.Next(5);` `int[] input = new int[] { r.Next(-7, -1),` `r.Next(-7, -1), r.Next(-7, -1),` `r.Next(-7, -1), r.Next(-7, -1) };` `for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(2, 10);` `Assert.AreEqual(n, MyTask.Calculate(input));` `}` `}` `[TestMethod]` `public void CheckZero()` `{` `int[] input = new int[0];` `Assert.AreEqual(0, MyTask.Calculate(input));` `}`
8	`[TestMethod]` `public void CheckNumberNegative()` `{` `int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };` `Assert.AreEqual(0, MyTask.Calculate(input1));` `int[] input2 = new int[] { -7, 5, 3, 6, 2 };` `Assert.AreEqual(1, MyTask.Calculate(input2));` `int[] input3 = new int[] { 10, 25, -4, -32, 19 };` `Assert.AreEqual(2, MyTask.Calculate(input3));` `}` `[TestMethod]` `public void CheckRandNumberNegative()` `{` `Random r = new Random();` `for (int i = 0; i < 100; i++)` `{` `int v = r.Next(-5, 7);` `int n = r.Next(5);` `int s = r.Next(5);` `int[] input = new int[] { r.Next(-2, 10),` `r.Next(2, 10), r.Next(2, 10),` `r.Next(2, 10), r.Next(2, 10) };` `for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(-7, -1);`

	Assert.AreEqual(n, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
9	[TestMethod] public void CheckMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4}; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(6, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(25, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
10	[TestMethod] public void CheckMaxIndex() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(2, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(3, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(1, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMaxIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) {

	`` int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckException() { int[] input = new int[0]; Assert.ThrowsException<Exception>(() => MyTask.Calculate(input)); }```
11	```[TestMethod] public void CheckAbsMax() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(8, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMax() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(7, 10); int p = r.Next(5); int[] input = new int[] { r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }```
12	```[TestMethod] public void CheckMin() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(-7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(-32, MyTask.Calculate(input3)); }```

	[TestMethod] public void CheckRandMin() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }
13	[TestMethod] public void CheckMinIndex() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(3, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(0, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(3, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMinIndex() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(p, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
14	[TestMethod] public void CheckMinAbs() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(2, MyTask.Calculate(input2));

```
    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(4, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMin()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(-5, 7);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10),
            r.Next(7, 10), r.Next(7, 10) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckMinusOne()
{
    int[] input = new int[0];
    Assert.AreEqual(-1, MyTask.Calculate(input));
}
```

15

```
[TestMethod]
public void CheckNumberPositive()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(6, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(4, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(3, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandNumberPositive()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
       int v = r.Next(-5, 7);
       int n = r.Next(5);
       int s = r.Next(5);
       int[] input = new int[] { r.Next(-7, -1),
            r.Next(-7, -1), r.Next(-7, -1),
            r.Next(-7, -1), r.Next(-7, -1) };
       for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(2, 10);
       Assert.AreEqual(n, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckException()
{
    int[] input = new int[0];
    Assert.ThrowsException<Exception>(() => MyTask.Calculate(input));
}
```

16

```
[TestMethod]
public void CheckNumberNegative()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(0, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(1, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(2, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandNumberNegative()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
      int v = r.Next(-5, 7);
      int n = r.Next(5);
      int s = r.Next(5);
      int[] input = new int[] { r.Next(-2, 10),
            r.Next(2, 10), r.Next(2, 10),
            r.Next(2, 10), r.Next(2, 10) };
      for (int k = 0; k < n; k++) input[(s + k) % input.Length] = r.Next(-7, -1);
      Assert.AreEqual(n, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckZero()
{
    int[] input = new int[0];
    Assert.AreEqual(0, MyTask.Calculate(input));
}
```

17

```
[TestMethod]
public void CheckMax()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4};
    Assert.AreEqual(8, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(6, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(25, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMax()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(7, 10);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7) };
        input[p] = v;
        Assert.AreEqual(v, MyTask.Calculate(input));
    }
}
```

```
[TestMethod]
public void CheckException()
{
    int[] input = new int[0];
    Assert.ThrowsException<Exception>(() => MyTask.Calculate(input));
}
```

18

```
[TestMethod]
public void CheckMaxIndex()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(2, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(3, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(1, MyTask.Calculate(input3));
}


[TestMethod]
public void CheckRandMaxIndex()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(7, 10);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(-5, 7),
        r.Next(-5, 7), r.Next(-5, 7),
        r.Next(-5, 7), r.Next(-5, 7) };
        input[p] = v;
        Assert.AreEqual(p, MyTask.Calculate(input));
    }
}

[TestMethod]
public void CheckZero()
{
    int[] input = new int[0];
    Assert.AreEqual(0, MyTask.Calculate(input));
}
```

19

```
[TestMethod]
public void CheckAbsMax()
{
    int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 };
    Assert.AreEqual(8, MyTask.Calculate(input1));

    int[] input2 = new int[] { -7, 5, 3, 6, 2 };
    Assert.AreEqual(7, MyTask.Calculate(input2));

    int[] input3 = new int[] { 10, 25, -4, -32, 19 };
    Assert.AreEqual(32, MyTask.Calculate(input3));
}

[TestMethod]
public void CheckRandMax()
{
    Random r = new Random();
    for (int i = 0; i < 100; i++)
    {
        int v = r.Next(7, 10);
        int p = r.Next(5);
        int[] input = new int[] { r.Next(-5, 7),
            r.Next(-5, 7), r.Next(-5, 7),
```

	r.Next(-5, 7), r.Next(-5, 7) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckZero() { int[] input = new int[0]; Assert.AreEqual(0, MyTask.Calculate(input)); }
20	[TestMethod] public void CheckMin() { int[] input1 = new int[] { 2, 3, 8, 1, 2, 4 }; Assert.AreEqual(1, MyTask.Calculate(input1)); int[] input2 = new int[] { -7, 5, 3, 6, 2 }; Assert.AreEqual(-7, MyTask.Calculate(input2)); int[] input3 = new int[] { 10, 25, -4, -32, 19 }; Assert.AreEqual(-32, MyTask.Calculate(input3)); } [TestMethod] public void CheckRandMin() { Random r = new Random(); for (int i = 0; i < 100; i++) { int v = r.Next(-5, 7); int p = r.Next(5); int[] input = new int[] { r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10), r.Next(7, 10) }; input[p] = v; Assert.AreEqual(v, MyTask.Calculate(input)); } } [TestMethod] public void CheckMinusOne() { int[] input = new int[0]; Assert.AreEqual(-1, MyTask.Calculate(input)); }

4А.4. Прочитать метаданные из файла заданного бинарного формата

Вам необходимо разработать программу, которая считывает заданное поле метаданных (см. варианты) из указанного бинарного формата и выводит его на экран.

Обрабатываемый файл указывается пользователем при запуске программы, которая может быть, на Ваш выбор, консольной или оконной.

Возможные варианты бинарных форматов:

Формат	Ссылки на описание
BMP	https://ru.wikipedia.org/wiki/BMP
PNG	https://habr.com/ru/post/130472/
JPEG	https://ru.wikipedia.org/wiki/JPEG, https://habr.com/ru/post/102521/
GIF	http://home.onego.ru/~chiezo/gif.htm, https://habr.com/ru/post/274917/
WAV	http://microsin.net/programming/pc/wav-format.html, https://audiocoding.ru/articles/2008-05-22-wav-file-structure/
AVI	https://ru.wikipedia.org/wiki/Audio_Video_Interleave

Для анализа содержимого файла вы можете воспользоваться бесплатным онлайн-ресурсом https://hexed.it/

В папке https://github.com/Nordth/istu-priklad-practic-2022/tree/master/media/day4a/Task_ReadBinaryFiles приведены примеры тестовых файлов, метаданные которых приведены в следующей таблице:

Файл	Поле метаданных	Значение
sun.bmp	ширина	64
	высота	58
	битность растра	24
bird.png	ширина	16
	высота	16
	битовая глубина цвета	8
	наличие альфа-канала	да
lenna.jpg	ширина	220
	высота	220
	количество каналов	3
cat.gif	ширина	64
	высота	67
	количество кадров	52
	наличие глобальной палитры	да
laugh.wav	число каналов	2
	частота дискретизации	48000
	количество бит в сэмпле	16
coding.avi	ширина	480
	высота	270
	число кадров в секунду	10

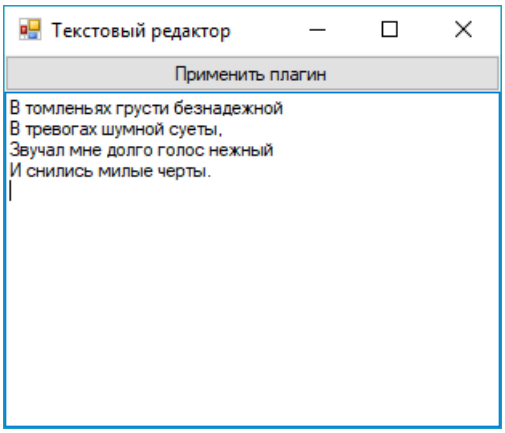
Ваша программа должна обрабатывать любые файлы заданного формата. В папке файлы приведены исключительно для примера

Варианты

№	Формат	Поле метаданных
1	BMP	ширина
2	BMP	высота
3	BMP	битность растра
4	PNG	ширина
5	PNG	высота
6	PNG	битовая глубина цвета
7	PNG	наличие альфа-канала
8	JPEG	ширина
9	JPEG	высота

10	JPEG	количество каналов
11	GIF	ширина
12	GIF	высота
13	GIF	количество кадров
14	GIF	наличие глобальной палитры
15	WAV	число каналов
16	WAV	частота дискретизации
17	WAV	количество бит в сэмпле
18	AVI	ширина
19	AVI	высота
20	AVI	число кадров в секунду

4А.5. Разработать плагин к текстовому редактору в виде DLL-библиотеки



Необходимо создать DLL-библиотеку, которая будет использоваться в качестве плагина к текстовому редактору. Плагин должен осуществлять заданную операцию над входным текстом и возвращать результирующий текст.

Текстовый редактор уже написан, его код и сборка лежат по адресу https://github.com/Nordth/istu-priklad-practic-2022/tree/master/media/day4a/Task_TextEditor. В данном проекте изменять ничего не нужно. Вам нужно создать **новый** проект типа «*Библиотека классов (.NET Framework)*». В этом проекте вам необходимо в пространстве имен *Task_TextEditor* создать класс *Plugin* с методом *Execute*, который принимает на вход строку и возвращает изменённый текст (см. пример).

После сборки вашего проекта запустите текстовый редактор, введите текст и нажмите кнопку «Применить плагин». Откроется окно выбора файла, выберите в нем вашу скомпилированную библиотеку (DLL-файл). После выбора создастся экземпляр вашего класса *Plugin*, вызовется метод *Execute* и результат запишется в исходное текстовое поле.

Пример

Преобразование: вырезать все пробелы

Код плагина:

```
namespace Task_TextEditor
{
    public class Plugin
    {
        public string Execute(string input)
        {
            return input.Replace(" ", "");
        }
    }
}
```

Варианты

№	Преобразование	№	Преобразование
1	Перевести все буквы первого слова каждой строке в верхний регистр	11	Удалить все слова размером меньше, чем из 5 букв
2	Перевернуть каждое слово в строке (абг деж -> гба жед)	12	Удалить все слова, в которых меньше 5 согласных
3	Удалить знаки препинания	13	Перевести в верхний регистр все гласные
4	Поменять местами слова в каждой паре слов: аб вг де жз -> вг аб жз де	14	Перевести в верхний регистр все согласные
5	Удалить все согласные	15	Удалить все слова, в которых больше 5 согласных
6	Перевернуть строки (абг деж -> жед гба)	16	Перевести первую букву каждого слова в верхний регистр
7	Перемешать в случайном порядке все слова в строке	17	Перевести в верхний регистр каждую вторую букву слов: абв гдеж -> аБв гДеЖ
8	Удалить все слова без согласных букв	18	Удалить все слова размером больше, чем из 5 букв
9	Удалить все слова без гласных букв	19	Удалить первое и последнее слова в строках
10	Удаление каждого второго слова	20	Удалить все гласные