ФИНАЛЬНЫЕ ЗАДАНИЯ НА ПРАКТИКУ

1. Прочесть несколько строк из консоли и вывести измененный согласно варианту текст

Пользователь вводит несколько строк текста, далее вводит пустую строку. Необходимо преобразовать текст согласно варианту и вывести его

Пример
Удаление каждого второго слова

```
Введите текст:
В томленьях грусти безнадежной
В тревогах шумной суеты,
Звучал мне долго голос нежный
И снились милые черты.

Результат:
В грусти
В шумной
Звучал долго нежный
И милые
```

Варианты

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

2. Разработка теста с вводом одного из вариантов ответа и сохранением результатов в файл

Вам необходимо разработать программу для тестирования знаний на заданную тему, посвященную прикладному программированию и языку C#. Тест должен состоять из 3-ех вопросов, формулировку которых вам нужно придумать самостоятельно.

Перед началом теста программа запрашивает имя пользователя.

Далее программа должна выводить вопросы **в случайном порядке** (для этого используйте класс *Random*). После вывода вопроса программа предлагает на выбор 4 варианта ответа, один из которых является правильным. Для того, чтобы ответить, пользователь вводит номер варианта и нажимает клавишу *Enter*. После этого программа должна вывести либо следующий вопрос (если они не кончились), либо результат тестирования. Результат тестирования выводится в виде числа процентов правильных ответов.

Кроме вывода на экран, результат должен записывать в файл *"results.txt"* в формате: `<Имя>: <Процент>`. Если пользователь вводит еще не существующее имя в файле, то его результат записывается в новой строке. Если имя уже есть, то строка с его результатом должна обновиться.

Пример
Тема: переменные

```
Введите ваше имя:
> Misha

Вопрос 1. Какое ключевое слово необходимо использовать для объявления
целочисленной переменной?
1) int
2) bool
3) integer
4) char
> 1
Вопрос 2. Какой оператор используется для присвоения переменной значения?
1) :=
2) <-
3) =
4) <<
> 2

Вопрос 3. Можно ли переменной типа int присвоить значение типа float?
1) Да
2) Да, если переменная объявлена с помощью var
3) Да, но значение будет автоматически приведено к целому числу
4) Нет
> 4

Ваш результат: 66.7%
```

В файле *results.txt* добавится/обновится строка:

```
Misha: 66.7%
```

Варианты

№	Преобразование	№	Преобразование
1	Стандартная библиотека C#	11	Ключевые слова C#
2	Функции и методы в C#	12	Механизм событий
3	Коллекции в C#	13	Обобщения (generics) в C#
4	UML-диаграммы	14	Классы в C#
5	Типы данных C#	15	Объектно-ориентированное программирование
6	Обработка файлов	16	Механизм исключений
7	Форматирование вывода в C#	17	Структуры в C#
8	Тестирование	18	Преобразование типов в C#
9	Массивы в C#	19	Этапы разработки ПО
10	Отличия C# от C	20	Операторы языка C#

3. Прочесть все файлы из указанной директории и сформировать сводный CSV-файл

Пользователь указывает путь до директории с CSV файлами (табличные данные, в которых каждая строка — это строка таблицы, а колонки разделены запятыми или точкой с запятой). Необходимо построить и сохранить CSV-файл отчет, согласно заданию. В отчете должен быть результат по каждому из входных файлов и общий результат.

Для получения списка файлов, см. *System.IO.Directory.GetFiles*

Пример

Запись: фамилия (строка), возраст (целое число), результат (вещественное число). Вывести среднее значение результата.

Входной файл jan.csv:

```
Петров, 20, 5.3
Иванов, 22, 7.6
Сидоров, 20, 5
```

Входной файл feb.csv:

```
Потемкин, 19, 2.3
Пушкин, 32, 9.6
```

Входной файл mar.csv:

```
Журавлев, 18, 7.3
Никитин, 33, 8.6
Жаров, 22, 3.5
Семечкин, 55, 4
```

Программа:

```
Введите путь к директории:
> data/dir/

Введите путь для сохранения файла:
> report.csv
```

Выходной файл:

```
jan.csv, 5.97
feb.csv, 5.95
mar.csv, 5.85
Общее, 5.91
```

Варианты

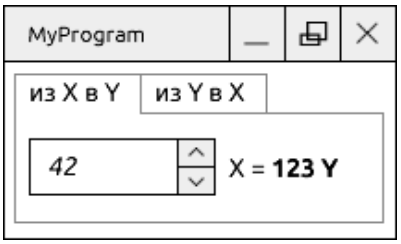
№	Запись	Отчет
1	Фамилия (строка), год поступления (целое число), средний балл (вещественное число)	Среднее значение *среднего балла*
2	Компания (строка), сумма поступлений в млн. руб. (вещественное число), сумма списаний в млн. руб. (вещественное число)	Сумма разниц между *суммами поступления* и *списания*
3	Название цеха (строка), план выпуска деталей (целое число), фактический выпуск деталей (целое число)	Сумма разниц между *планом выпуска деталей* и *фактическим выпуском*
4	Фамилия (строка), оценка за теорию (целое число), оценка за практику (целое число)	Среднее значение *всех оценок*
5	Автомобильный номер (строка), год выпуска (целое число), пробег в км (целое число)	Сумма значений *пробега*
6	Номер телефона (строка), имя оператора (строка), баланс в копейках (целое число)	Сумма *балансов*
7	Тема письма (строка), адресат (строка), есть ли вложения (логический тип), число слов (целое число)	Среднее *число слов*
8	Производитель (строка), объем выпуска (вещественное число), средняя цена (вещественное число)	Среднее значение *средней цены*
9	Адрес отправления (строка), адрес доставки (строка), вес (вещественное число)	Сумма *весов*
10	Дисциплина (строка), номер курса (целое число), количество часов (целое число)	Среднее *количество часов*
11	Фамилия (строка), номер группы (строка), номер в группе (целое число), число выполненных заданий (целое число)	Среднее число *выполненных заданий*
12	Город (строка), улица (строка), номер дома (число), номер этажа (целое число)	Среднее значение *номера этажа*
13	Фамилия (строка), рост (вещественное число), вес (вещественное число)	Среднее значение *роста*
14	Название товара (строка), количество на складе (целое число), количество зарезервированных (целое число)	Сумма разниц между *числом товара на складе* и *числом зарезервированных товаров*
15	Название материала (строка), объем (вещественное число), вес (вещественное число)	Сумма *объемов*
16	Название (строка), число сезонов (целое число), год выпуска первого сезона (целое число)	Среднее *число сезонов*
17	Фамилия (строка), число ролей (целое число), гонорар в млн. руб. (вещественное число)	Среднее значение *гонорара*

18	Номер заказа (строка), описание (строка), выполнен или нет (логический тип), сумма заказа (целое число)	Сумма *сумм заказов*
19	Адрес сайта (строка), число посетителей (целое число), число уникальных посетителей (целое число)	Среднее *число уникальных посетителей*
20	Фамилия (строка), должность (строка), оклад в руб (целое число)	Сумма *окладов*

4. Создать программу-калькулятор пересчета единиц измерения

Создать форму для пересчета единиц измерения, состоящую из двух вкладок (*TabControl*), определяющих направление перевода. Каждая вкладка отвечает за свое направление перевода и содержит следующие элементы:

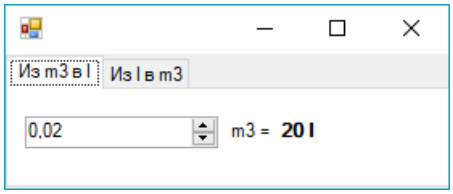
- поле ввода входного значения
- блок вывода результата



При вводе входного значения, результат пересчитывается автоматически, без дополнительных действий пользователя. Входное значение должно быть синхронизировано между вкладками, т.е. сохраняться при переключении.

В блоке вывода результат должен быть выделен жирным шрифтом.

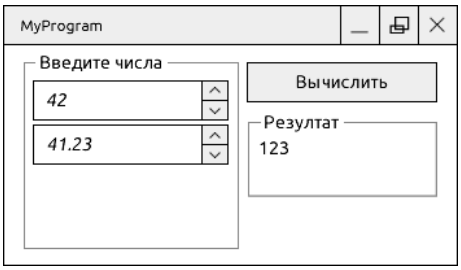
Пример
Перевод куб. метров в литры (1 m3 = 1000 l) и наоборот



Варианты

№	Перевод	№	Перевод	№	Перевод	№	Перевод
1	1 $ = 67.2 rub	6	1 h = 60 min	11	1 gal = 3.7854 L	16	1 ft = 0.3048 m
2	1 c = 100 kg	7	1 ha = 0.01 km2	12	1 ozt = 31.103 g	17	1 y = 12 month
3	1 km = 1000 m	8	1 min = 60 s	13	1 s = 1000 ms	18	1 KB = 1024 B
4	1 d = 24 h	9	1 m = 100 cm	14	1 y = 365 d	19	1 d = 2.54 cm
5	1 eur = 77.1 rub	10	1 p = 0.4536 kg	15	1 kg = 1000 g	20	1 hf = 735.5 W

5. Создать программу, выполняющую заданное действие с набором чисел, вводимых в динамически создаваемые поля



Создать форму, содержащую следующие элементы:

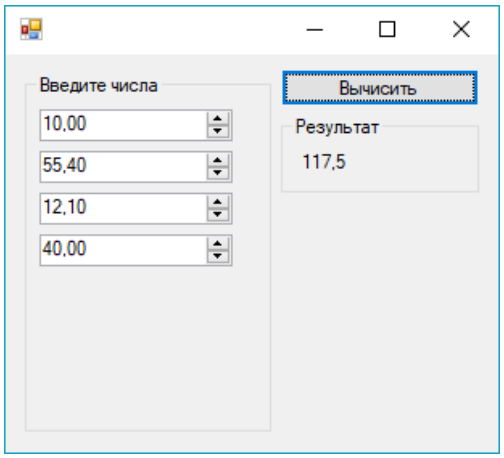
- блок ввода чисел: *TableLayoutPanel*, в котором динамически добавляются элементы *NumericUpDown* со свойствами *Increment* = 0.01 и *DecimalPlaces* = 2
- кнопка «Вычислить»
- блок с результатом

Изначально в блоке ввода чисел должен быть только один элемент *NumericUpDown*. Если фокус находится на этом элементе и пользователь нажимает клавишу «Enter», ниже должно появиться следующее поле ввода и фокус должен переместиться на него. Если фокус находится на одном из элементов блока ввода и пользователь нажимает клавишу «Delete», этот элемент должен удалиться (если он не последний) и фокус должен переместиться на элемент выше (или ниже, если это первый элемент).

При нажатии на кнопку «Вычислить» программа должна выполнить требуемую операцию со всеми числами из блока ввода и вывести полученное значение в блоке результата

Пример

Просуммировать числа

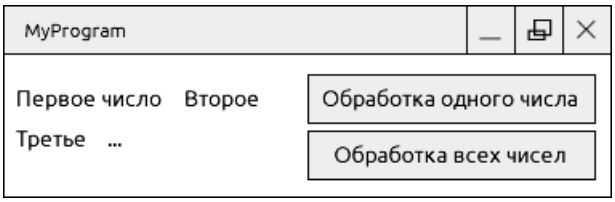


Варианты

№	Операция	№	Операция
1	Перемножить все положительные числа	11	Просуммировать модули всех чисел
2	Просуммировать все нечетные числа	12	Вычислить среднее из квадратов чисел
3	Просуммировать все четные числа	13	Просуммировать квадраты всех чисел
4	Перемножить дробные части всех чисел	14	У каждого второго числа поменять знак и просуммировать все числа

5	Разделить все числа на первое из чисел и вычислить сумму результатов	15	У каждого второго числа поменять знак и посчитать среднее
6	Умножить числа на их порядковые номера и сложить результаты	16	Перемножить все отрицательные числа
7	Извлечь квадратный корень из суммы квадратов чисел	17	Вычислить среднее значение
8	Просуммировать дробные части всех чисел	18	Просуммировать все числа, которые больше первого из чисел
9	Умножить числа на их порядковые номера и вычислить среднее	19	Округлить числа и сложить результаты
10	Просуммировать все числа, которые меньше, чем предыдущее число	20	Просуммировать все числа, которые больше, чем предыдущее число

6. Создать программу для обработки списка случайных целых чисел



Создать форму состоящую из:

- блока случайных чисел (*FlowLayoutPanel*)
- блока команд

Блок чисел изначально наполняется случайным количеством (от 5 до 10) случайных чисел (в диапазоне от -100 до 100). Каждое число представляется в виде элемента *Label* и добавляется в *FlowLayoutPanel* с помощью:

```
<Экземпляр FlowLayoutPanel>.Controls.Add(<Элемент>
```

Блок команд состоит из двух кнопок: команда обработки одного числа и команда обработки всех чисел.

Кнопка командыы обработки одного числа изначально выключена. Для того, чтобы воспользоваться этой командой, необходимо выбрать число, нажав на него кнопкой мыши. Выбранное число должно выделиться жирным. Если пользователь нажмет на другое число, то выделение должно перейти на него, т.е. одновременно может быть выделено только одно число. После выполнения команды обработки, выделение снимается. Кроме этого, командой также можно воспользоваться **дважды кликнув** на число, которое требуется обработать (событие *DoubleClick*)

Команда обработки всех чисел должна выдавать результат в стандартном всплывающем окне: *MessageBox.Show(…)*

Список возможных команд обработки одного числа

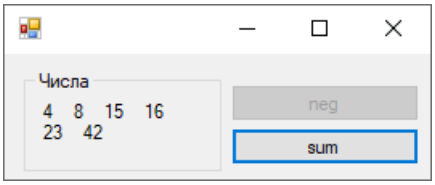
- *neg* – меняет знак у числа
- *delete* –удаляет число из списка
- *copy* – добавляет в конец списка копию числа
- *twice* – увеличивает число вдвое
- *rand* – менят значение числа на другое случайное в диапазоне от -100 до 100

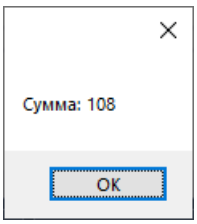
Список возможных команд обработки всех чисел:

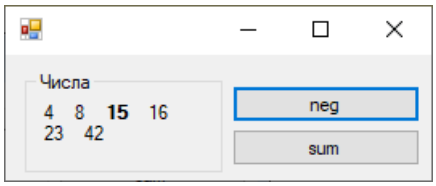
- *sum* – выводит сумму чисел
- *sumeven* – выводит сумму четных чисел
- *sumodd* – выводит сумму нечетных чисел
- *max* – выводит максимальное из чисел
- *min* – выводит минимальное из чисел
- *avg* – выводит среднее из чисел
- *disp* – выводит разницу между максимальным и минимальным числами

Пример

Команды: neg и sum







Варианты

№	Одно число	Все числа	№	Одно число	Все числа
1	delete	min	11	neg	disp
2	twice	sum	12	delete	sum
3	delete	min	13	twice	sumeven
4	twice	avg	14	copy	disp
5	copy	max	15	neg	min
6	rand	sumodd	16	copy	sumodd
7	twice	avg	17	rand	sumeven
8	neg	avg	18	neg	sumodd
9	copy	sumeven	19	delete	max
10	rand	sum	20	rand	max

7. 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));
}
```

8. Сконвертировать входной CSV-файл в бинарный файл заданного формата

Вы должны разработать программу (консольную или оконную, на Ваш выбор), которая преобразует входной CSV-файл в бинарный файл заданного формата. Бинарный файл должен состоять из трех секций:

1. Заголовок
2. Оглавление
3. Данные

Секция «Заголовок» состоит из 6 байт:

Смещение	**Размер**	**Описание**
00-01	2	Два байта 0x50 (сигнатура)
02-05	4	Типы полей вашей структуры по варианту. Первый байт задает тип первого поля, второй – второго и т.д. Возможные значения типа: • 0x49 – целый

		• 0x44 – вещественный • 0x42 – логический • 0x53 – строковый • 0x00 – поля нет

Секция «Оглавление» состоит из двух элементов:

- количество записей в файле (4 байта)
- список, в котором для каждой записи указано смещение, по которому находятся данные этой записи. Смещения указываются относительно начала секции «Данные» и занимают 4 байта. Таким образом, если в файле N записей, размер списка – 4N байт

В секции «Данные» последовательно располагается содержимое записей. В зависимости от типа, поля в записи занимают разный объем памяти

- *Целое поле* занимает 4 байта
- *Вещественное поле* – 8 байт
- *Логическое поле* – 1 байт (0x01 – true, 0x00 – false)
- *Строковое поле* записывается в виде двух значений: размер строки в байтах (2 байта) и сама строка (в кодировке UTF-8)

Порядок записи байт в числах – от младшего к старшему (little-endian)

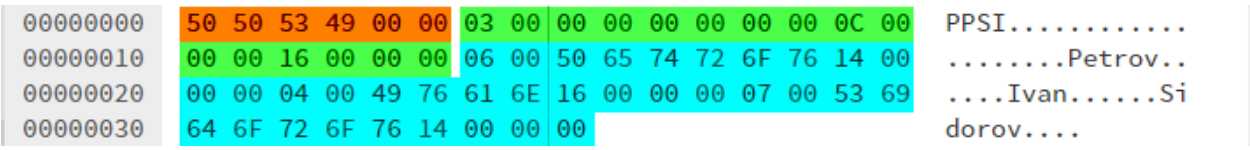
Пример

Запись: фамилия (строка), возраст (целое число)

Входной файл:

```
Petrov, 20
Ivan, 22
Sidorov, 20
```

Выходной бинарный файл (в шестнадцатеричном представлении)



Секция «Заголовок» обозначена оранжевым цветом:

Фрагмент	**Описание**
50 50	Сигнатура файла
53 49 00 00	Два поля в структуре: 1) строковое, 2) целочисленное

Секция «Оглавление» обозначена зеленым цветом

Фрагмент	**Описание**
03 00 00 00	Количество записей в файле (3)
00 00 00 00	Смещение первой записи относительно секции «Данные» (0)
0C 00 00 00	Смещение второй записи относительно секции «Данные» (12)
16 00 00 00	Смещение третьей записи относительно секции «Данные» (22)

Секция «Данные» обозначена голубым цветом

Фрагмент	**Описание**
06 00 50 65 74 72 6F 76	Поле «Фамилия» записи №1 (длина 6 байт, “Petrov”)
14 00 00 00	Поле «Возраст» записи №1 (20)
04 00 49 76 61 6E	Поле «Фамилия» записи №2 (длина 4 байта, “Ivan”)
16 00 00 00	Поле «Возраст» записи №2 (22)
07 00 53 69 64 6F 72 6F 76	Поле «Фамилия» записи №3 (длина 7 байт, “Sidorov”)
14 00 00 00	Поле «Возраст» записи №3 (20)

Варианты

№	**Запись**
1	Фамилия (строка), номер группы (строка), номер в группе (целое число), число выполненных заданий (целое число)
2	Тема письма (строка), адресат (строка), есть ли вложения (логический тип), число слов (целое число)
3	Фамилия (строка), число ролей (целое число), гонорар в млн. руб. (вещественное число)
4	Производитель (строка), объем выпуска (вещественное число), средняя цена (вещественное число)
5	Адрес отправления (строка), адрес доставки (строка), вес (вещественное число)
6	Название товара (строка), количество на складе (целое число), количество зарезервированных (целое число)
7	Город (строка), улица (строка), номер дома (число), номер этажа (целое число)
8	Номер заказа (строка), описание (строка), выполнен или нет (логический тип), сумма заказа (целое число)
9	Номер телефона (строка), имя оператора (строка), баланс в копейках (целое число)
10	Название материала (строка), объем (вещественное число), вес (вещественное число)
11	Фамилия (строка), год поступления (целое число), средний балл (вещественное число)
12	Название (строка), число сезонов (целое число), год выпуска первого сезона (целое число)
13	Фамилия (строка), оценка за теорию (целое число), оценка за практику (целое число)
14	Дисциплина (строка), номер курса (целое число), количество часов (целое число)
15	Адрес сайта (строка), число посетителей (целое число), число уникальных посетителей (целое число)
16	Фамилия (строка), рост (вещественное число), вес (вещественное число)
17	Компания (строка), сумма поступлений в млн. руб. (вещественное число), сумма списаний в млн. руб. (вещественное число)
18	Автомобильный номер (строка), год выпуска (целое число), пробег в км (целое число)
19	Название цеха (строка), план выпуска деталей (целое число), фактический выпуск деталей (целое число)
20	Фамилия (строка), должность (строка), оклад в руб (целое число)