1.

Попарно должны быть перегружены следующие операторы (*укажите все верные ответы***):**
1) **true** и **false**. 2) **explicit** и **implicit**. 3) + и -. 4) ++ и --. 5) < и >.

2.

Могут быть перегружены следующие операторы (попарную перегрузку не учитывать) (*укажите все верные ответы***):**
1) checked. 2) typeof. 3) true. 4) false. 5) =.

3.

Верно, что перегрузка операторов (*укажите все верные ответы***):**
1) Может объявляться только с модификаторами **public** и **static**. 2) Может быть осуществлена для любых типов. 3) Присутствует у зарезервированных значимых типов. 4) Поддерживают параметры, передаваемые как по значению, так и по ссылке. 5) Доступна для перегрузки операции приведения типов.

4.

В результате выполнения фрагмента программы:

```
using System;

class Point {
    public int x;
    public int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    public override string ToString() {
        return $"{x}{y}";
    }

    public static Point operator +(Point arg, Point arg2) {
        Point obj = new Point(0, 0);
        obj.x = arg.x + arg2.x;
        obj.y = arg.y + arg2.y;
        return obj;
    }
}

class Program {
    static void Main() {
        Point arg1 = new Point(5, 6);
        Point arg2 = new Point(2, 9);
        Console.WriteLine(arg1 + arg2);
    }
}
```

на экран будет выведено:

5.

В результате выполнения фрагмента программы:

```
using System;

class Class {
    internal int x = 8;
    internal int y = 5;
    internal int z = 3;

    struct Struct {
        internal int x;
        internal int y;
        internal int z;

        public Struct(int x, int y, int z) {
            this.x = x;
            this.y = y;
            this.z = z;
        }

        public static Struct operator *(Class obj1, Struct
obj2) {
            Struct obj = new Struct();
            obj.x = obj1.x * obj2.x;
            obj.y = obj1.y * obj2.y;
            obj.z = obj1.z * obj2.z;
            return obj;
        }
    }

    static void Main() {
        Struct obj1 = new Struct(10, 20, 30);
        Class obj2 = new Class();
        Console.WriteLine((obj2 * obj1 * obj2).z);
    }
}
```

на экран будет выведено:

Примечание:

Если возникнет ошибка компиляции, введите: ***

Если ошибок и исключений нет, но на экран не выведется ничего, введите: ---

Если возникнет ошибка исполнения или исключение, введите: +++

6.

Выберите все верные утверждения:

1) Enumerable – статический класс.
2) Интерфейс IEnumerable имеет как определённую реализацию, так и обобщённую реализацию.
3) При инициализации анонимной переменной обязательно ключевое слово **new**.
4) Все параметры анонимной переменной являются параметрами «только для чтения».
5) Анонимная переменная – объект ссылочного типа.

7.

Интерфейс IEnumerator<T> декларирует следующие члены:

1) void Reset();
2) T Current;
3) object Current;
4) void Dispose();
5) void GetNext();

8.

В результате выполнения фрагмента программы:

```
using System;
using System.Collections;

class Program : IEnumerator {
    int[] args = new int[] { 4, 5, 39, 10, 11, 34, 29 };
    int pos = -1;

    public object Current => args[pos];

    static void Main() {
        Program obj = new Program();
        foreach (var item in obj) {
            Console.Write(item);
        }
    }

    public IEnumerator GetEnumerator() {
        return new Program();
    }

    public bool MoveNext() {
        pos += 2;
        if (pos < args.Length) return true;
        if ((pos ^ 4) == 3) {
            Reset();
            return true;
        }
        return false;
    }

    public void Reset() {
        pos = 2;
    }
}
```

на экран будет выведено:

9.

В результате выполнения фрагмента программы:

```
using System;
using System.Collections;

class MyClass {
    public IEnumerable BlackAndWhite(int start, int end) {
        for (int i = start; i <= end; i++) {
            yield return i * 5 / 6 % 2;
        }
    }
}

class Program {
    static void Main() {
        MyClass mc = new MyClass();
        foreach (object obj in mc.BlackAndWhite(3, 9))
            Console.Write(obj);
    }
}
```

на экран будет выведено:

10.

В результате выполнения фрагмента программы:

```
using System;
using System.Collections;

class Program {
    static void Main() {
        int[] MyArray = { 10, 11, 12, 13 };
        IEnumerator ie = MyArray.GetEnumerator();
        while (ie.MoveNext()) {
            int i = (int)ie.Current;
            if (!ie.MoveNext()) ie.Reset();
            Console.Write(i);
        }
    }
}
```

на экран будет выведено:

Примечание:

Если возникнет ошибка компиляции, введите: ***

Если ошибок и исключений нет, но на экран не выведется ничего, введите: ---

Если возникнет ошибка исполнения или исключение, введите: +++

1	15
2	34
3	15
4	715
5	***
6	12345
7	1234
8	51034391129
9	0101101
10	1012