1.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        try {
            object o = null;
            Console.Write(o.GetType());
        }
        catch (SystemException) {
            Console.Write("System");
        }
        catch (NullReferenceException) {
            Console.Write("Null");
        }
        catch (Exception) {
            Console.Write("Exception");
        }
        finally {
            Console.Write("Final");
        }
    }
}
```

на экран будет выведено:

2.

Выберите верные утверждения (*укажите все верные ответы***):**

1) В операторе try-catch-finally обязателен блок finally.
2) В операторе try-catch-finally обязателен блок catch.
3) Оператор throw без аргумента исключения пробрасывает пойманное исключение дальше.
4) NullReferenceException унаследован от ApplicationException.
5) Исключение – тип значения.

3.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        try {
            object o = null;
            return;
            Console.Write(o.GetType());
        }
        catch (NullReferenceException) {
            Console.Write("Null");
        }
        finally {
            Console.Write("Final");
        }
    }
}
```

на экран будет выведено:

4.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        try {
            object o = null;
            object o2 = 5;
            Console.Write(o2.Equals(o));
        }
        catch (NullReferenceException) {
            Console.Write("Null");
        }
        finally {
            Console.Write("Final");
        }
    }
}
```

на экран будет выведено:

5.

Выберите верные утверждения (*укажите все верные ответы***):**

1) На отлов исключений можно наложить условия их перехвата с помощью ключевого слова where.
2) Оператор throw нельзя использовать без аргумента-исключения.
3) Оператор throw можно использовать с типами значений.
4) Все исключения унаследованы от класса Exception.
5) В классе Exception – 4 конструктора, среди которых есть беспараметрический конструктор.

6.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        int[] args = new int[1];
        try {
            for (int i = 0; i <= args.Length; i++) {
                args[i] = i + 1;
            }
        }
        catch (IndexOutOfRangeException) {
            if (args[0] == 0) throw;
            Console.Write(1);
        }
        catch (SystemException) {
            if (args.IsFixedSize) throw;
            Console.Write(2);
        }
        catch (Exception) {
            Console.Write(3);
        }
        finally {
            Console.Write(5);
        }
    }
}
```

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: +++

7.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        int i = 0;
        Label5:
        try {
            i++;
            int a = 0;
            int b;
            if(i == 5)
            goto Label1;
            else b = 5 / a;
        }
        catch (DivideByZeroException) when (i == 1) {
            Console.Write(1);
            goto Label5;
        }
        catch (DivideByZeroException) when (i == 2) {
            Console.Write(2);
            goto Label5;
        }
        catch (DivideByZeroException) when (i == 3) {
            Console.Write(3);
            goto Label5;
        }
        catch (DivideByZeroException) when (i > 3) {
            Console.Write(4);
            goto Label5;
        }
        finally {
            Console.Write(0);
        }
        Label1:
        Console.Write(5);
    }
}
```

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: +++

8.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        try {
            object o = null;
            o.Equals(null);
        }
        catch (ArgumentNullException) {
            Console.Write(0);
        }
        catch (NullReferenceException) {
            Console.Write(1);
        }
        Console.Write(2);
    }
}
```

на экран будет выведено:

9.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        string s = "12345";
        int arg;
        try {
            int.Parse(string.Concat(s[0] + s[1]));
        }
        catch (FormatException) {
            Console.Write("Format" + s);
        }
        Console.Write("String" + s);
    }
}
```

на экран будет выведено:

10.

В результате выполнения фрагмента программы:

```
using System;

class Program {
    static void Main() {
        try {
            Console.Write("Try");
            Print();
        }
        catch (StackOverflowException) {
            Console.Write("Catch");
        }
        finally {
            Console.Write("Finally");
        }
    }

    public static void Print() {
        Console.Write("Exception");
        Print();
    }
}
```

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	***
2	3
3	Final
4	FalseFinal
5	245
6	15
7	1020304005
8	12
9	String12345
10	+++