

Output Reasoning Type Questions

What will be the output of the following C# codes? Why?

1.

```
using System;
class Program
{
    public void Main(string[] args)
    {
        int a = 0;
        Console.WriteLine(a++);
    }
}
```

2.

```
class Program
{
    public static void Main(string[] args)
    {
        int a = 0;
        int b = a++;
        Console.WriteLine(a++.ToString(), ++a, -a++);
        Console.WriteLine((a++).ToString() + (-a++).ToString());
        Console.WriteLine(~b);
    }
}
```

3.

```
using System;
/*you can also write top level code outside of a class. C# takes care of this by providing internal entry point Main*/

Console.WriteLine("int x = 3;");
Console.WriteLine("int y = 2 + ++x;");

int x = 3; //default visibility is 'internal' (outside a class)
int y = 2 + ++x;
Console.WriteLine($"x = {x} and y = {y}");

Console.WriteLine("x = 3 << 2;");
Console.WriteLine("y = 10 >> 1;");

x = 3 << 2;
y = 10 >> 1;
Console.WriteLine($"x = {x} and y = {y}");

x = ~x;
y = ~y;
Console.WriteLine($"x = {x} and y = {y}");
```

4.

```
using System;
public class Program
{
    static void Main()
    {
        try
        {
            int i=int.MaxValue;
            Console.WriteLine(-(i+1)-i);
            for(i=0; i<=int.MaxValue;i++); //note semicolon here
            Console.WriteLine("Program ended!");
        }
        catch(Exception ex)
        {
            Console.WriteLine(ex.ToString());
        }
    }
}
```

5.

```
using System;
public class Program
{
    static void Main(string[] args)
    {
        Main(["CS202"]);
    }
}
```

6.

```
using System;
int a = 3;
int b = a++;
Console.WriteLine($"a is {a++}, b is {--b}");

int c = 3;
int d = ++c;
Console.WriteLine($"c is {-c--}, d is {~d}");
```

7.

```
using System;
class Program
{
    int age;
    Program() => age=age==0?age+1:age-1;
    static void Main()
    {
        int k = "010%".Replace('0','%').Length;
        Console.Write "[" + (k<<+new Program().age).ToString() + "]";
        Console.Write "[" + "010%".Split('1')[1][0] + "]";
        Console.Write "[" + "010%".Split('0')[1][0] + "]";
        Console.Write "[" + int.Parse(Convert.ToString("123".ToCharArray()[~-1])) + "]";
    }
}
```

8.

```
using System;

class Program
{
    static void Main()
    {
        int[] nums = {0, 1, 0, 3, 12};
        int pos = 0;

        for (int i = 0; i < nums.Length; i++)
        {
            if (nums[i] != 0)
            {
                int temp = nums[pos];
                nums[pos] = nums[i];
                nums[i] = temp;
                pos++;
            }
        }

        Console.WriteLine(string.Join(" ", nums));
    }
}
```

9.

```
using System;
class Program
{
    int f;
    public static void Main(string[] args)
    {
        Console.WriteLine("run 1");
        Program p = new Program(new int[]{"0".Length});
        for (int i = 0, _ = i; i < 5 && ++p.f >= 0; i++, Console.WriteLine(p.f++));
        {
            for (;p.f == 0;);
            {
                Console.WriteLine(p.f);
            }
        }
    }
}
```

```

        Console.WriteLine("\nrun 2");
        p = new Program(p.f);
        Console.WriteLine(p.f);

        Console.WriteLine("\nrun 3");
        p = new Program();
        Console.WriteLine(p.f);
    }
    Program() => f = 0;
    Program(int x) => f=x;
}

```

10.

```

public class A
{
    public virtual void f1()
    {
        Console.WriteLine("f1");
    }
}
public class B:A
{
    public override void f1() => Console.WriteLine("f2");
}

class Program
{
    static int i=0;
    public event funcPtr handler;
    public delegate void funcPtr();
    public void destroy()
    {
        if (i == 6)
            return;
        else
        {
            Console.WriteLine(i++);
            destroy();
        }
    }
    public static void Main(string[] args)
    {
        Program p = new Program();
        p.handler += new funcPtr((new A()).f1);
        p.handler += new funcPtr((new B()).f1);
        p.handler();
        p.handler -= new funcPtr((new B()).f1);
        p.handler -= new funcPtr((new A()).f1);
        p?.destroy(); //check herea about ?. operator
        p = null;
        i = -6;
        p?.destroy();
        (new Program())?.destroy();
    }
}

```

11.

```

public class Institute
{
    internal int i = 7;
    public Institute()
    {
        Console.Write("1");
    }
    public virtual void info()
    {
        Console.Write("2");
    }
}
public class IITGN:Institute
{
    public int i = 8;
    public IITGN()
    {
        Console.Write("3");
    }
    public IITGN(int i)
    {

```

```

        Console.Write("4");
    }
    public override void info()
    {
        Console.Write("5");
    }
}
class Program
{
    public static void Main(string[] args)
    {
        Console.Write("6");
        Institute ins1 = new Institute();
        ins1.info();
        IITGN ab101 = new IITGN(3);
        ab101 = new IITGN();
        ab101.info();
        Console.WriteLine();
        Console.WriteLine(ab101.i);
        Console.WriteLine(~(((Institute)ab101).i));
    }
}

```

12.

```

using System;
public class Program
{
    public delegate void mydel();
    public void fun1()
    {
        Console.WriteLine("fun1()");
    }
    public void fun2()
    {
        Console.WriteLine("fun2()");
    }
    public static void Main(string[] args)
    {
        Program p = new Program();

        mydel obj1 = new mydel(p.fun1);
        obj1 += new mydel(p.fun2);
        Console.WriteLine("run 1");
        obj1();

        mydel obj2 = new mydel(p.fun2);
        obj2 += new mydel(p.fun1);
        Console.WriteLine("run 2");
        obj2();

        obj2 -= p.fun2;
        Console.WriteLine("run 3");
        obj2();
    }
}

```

13.

```

using System;
using System.Collections9;
public class Program
{
    int x;
    public static void Main(string[] args)
    {
        ArrayList10 L=new ArrayList();
        L.Add(new Program());
        L.Add(new Program());
        for (int i=0;i<L.Count;i++)
            Console.WriteLine(++((Program)L[i]).x);

        L[0]=L[1];
        ((Program)L[0]).x = 202;

        for (int i=0;i<L.Count;i++)
            Console.WriteLine(((Program)L[i]).x);

        ((Program)L[0]).x = 111;
        L.RemoveAt(0);
    }
}

```

14.

```
        Console.WriteLine(L.Count);
        Console.WriteLine(((Program)L[0]).x);
    }
}
```

15.

```
using System;

delegate int Calc(int x, int y);

class Program
{
    static int Add(int a, int b) { Console.Write("A"); return a + b; }
    static int Mul(int a, int b) { Console.Write("M"); return a * b; }
    static int Sub(int a, int b) { Console.Write("S"); return a - b; }

    static void Main()
    {
        Calc c = Add;
        c += Mul;
        c += Sub;
        c -= Add;
        int res = c(2, 3);
        Console.Write(": " + res);
    }
}
```

16.

```
using System;

delegate void ActionHandler(ref int x);

class Program
{
    static void Inc(ref int a) { a += 2; Console.Write("I" + a + " "); }
    static void Dec(ref int a) { a--; Console.Write("D" + a + " "); }

    static void Main()
    {
        int val = 3;
        ActionHandler act = Inc;
        act += Dec;
        act(ref val);
        Console.Write("F" + val);
    }
}
```

```
using System;

class LimitEventArgs : EventArgs
{
    public int CurrentValue { get; }
    public LimitEventArgs(int val) => CurrentValue = val;
}

class Counter
{
    public event EventHandler<LimitEventArgs> LimitReached;
    public event EventHandler<LimitEventArgs> MilestoneReached;

    private int value = 0;

    public void Increment()
    {
        value++;
        Console.Write(">" + value);

        // Fire Milestone event every 2nd increment
        if (value % 2 == 0)
            MilestoneReached?.Invoke(this, new LimitEventArgs(value));

        // Fire Limit event every 3rd increment
        if (value % 3 == 0)
            LimitReached?.Invoke(this, new LimitEventArgs(value));
    }
}
```

```

class Program
{
    static void Main()
    {
        Counter c = new Counter();

        // Subscribers for LimitReached
        c.LimitReached += (s, e) => Console.WriteLine("[L] + e.CurrentValue + "]");
        c.LimitReached += (s, e) => Console.WriteLine("(Reset)");

        // Subscribers for MilestoneReached
        c.MilestoneReached += (s, e) =>
        {
            Console.WriteLine("[M] + e.CurrentValue + "]");
            if (e.CurrentValue == 4)
                Console.WriteLine("{Alert}");
        };

        for (int i = 0; i < 6; i++)
            c.Increment();
    }
}

```

17.

```

using System;

class TemperatureEventArgs : EventArgs
{
    public int OldTemperature { get; }
    public int NewTemperature { get; }

    public TemperatureEventArgs(int oldTemp, int newTemp)
    {
        OldTemperature = oldTemp;
        NewTemperature = newTemp;
    }
}

class TemperatureSensor
{
    public event EventHandler<TemperatureEventArgs> TemperatureChanged;

    private int temperature = 25;

    public void UpdateTemperature(int newTemp)
    {
        int oldTemp = temperature;
        temperature = newTemp;

        if (Math.Abs(newTemp - oldTemp) > 5)
        {
            TemperatureChanged?.Invoke(this, new TemperatureEventArgs(oldTemp, newTemp));
        }
    }
}

class Program
{
    static void Main()
    {
        TemperatureSensor sensor = new TemperatureSensor();

        sensor.TemperatureChanged += (s, e) =>
            Console.WriteLine($"Temperature changed from {e.OldTemperature}°C to {e.NewTemperature}°C");

        sensor.TemperatureChanged += (s, e) =>
        {
            if (Math.Abs(e.NewTemperature - e.OldTemperature) > 10)
                Console.WriteLine(" Warning: Sudden change detected!");
        };

        sensor.UpdateTemperature(28);
        sensor.UpdateTemperature(30);
        sensor.UpdateTemperature(46);
        sensor.UpdateTemperature(52);
    }
}

```

18.

```

using System;

class NotifyEventArgs : EventArgs
{
    public string Message { get; }
    public NotifyEventArgs(string msg) => Message = msg;
}

class Notifier
{
    public event EventHandler<NotifyEventArgs> OnNotify;

    public void Trigger(string msg)
    {
        Console.WriteLine("[Start]");
        OnNotify?.Invoke(this, new NotifyEventArgs(msg));
        Console.WriteLine("[End]");
    }
}

class Program
{
    static void Main()
    {
        Notifier n = new Notifier();

        n.OnNotify += (s, e) =>
        {
            Console.WriteLine("{ " + e.Message + " }");
        };

        n.OnNotify += (s, e) =>
        {
            Console.WriteLine("(Nested)");
            if (e.Message == "Ping")
                ((Notifier)s).Trigger("Pong");
        };

        n.Trigger("Ping");
    }
}

```

19.

```

using System;

class AlertEventArgs : EventArgs
{
    public string Info { get; }
    public AlertEventArgs(string info) => Info = info;
}

class Sensor
{
    public event EventHandler<AlertEventArgs> ThresholdReached;

    public void Check(int value)
    {
        Console.WriteLine("[Check]");
        if (value > 50)
            ThresholdReached?.Invoke(this, new AlertEventArgs("High"));
        Console.WriteLine("[Done]");
    }
}

class Program
{
    static void Main()
    {
        Sensor s = new Sensor();

        s.ThresholdReached += (sender, e) =>
        {
            Console.WriteLine("{ " + e.Info + " }");
            if (e.Info == "High")
                ((Sensor)sender).Check(30);
        };
    }
}

```

```

        s.ThresholdReached += (sender, e) =>
            Console.WriteLine("Alert");

        s.Check(80);
    }
}

```

20. Does this code compile? If not, identify the error and explain why. If it does, what would print?

```

public delegate void AuthCallback(bool validUser);
public static AuthCallback loginCallback = Login;
public static void Login()
{
    Console.WriteLine("Valid user!");
}

public static void Main(string[] args)
{
    loginCallback(true);
}

```

- 21.

```

using System;

delegate void Notify(string msg);

class Program
{
    static void Main()
    {
        Notify handler = null;

        handler += (m) => Console.WriteLine("A: " + m);
        handler += (m) => Console.WriteLine("B: " + m.ToUpper());

        handler("hello");

        handler -= (m) => Console.WriteLine("A: " + m);
        handler("world");
    }
}

```

- 22.

```

using System;

class Program
{
    static string txtAge;
    static DateTime selectedDate;
    static int parsedAge;

    static void Main(string[] args)
    {
        try
        {
            Console.WriteLine(txtAge == null ? "txtAge is null" : txtAge);

            Console.WriteLine(selectedDate == default(DateTime)
                ? "selectedDate is default"
                : selectedDate.ToString());

            if (string.IsNullOrEmpty(txtAge))
            {
                Console.WriteLine("txtAge is null or empty, cannot parse");
            }
            else
            {
                parsedAge = int.Parse(txtAge);
                Console.WriteLine($"Parsed Age: {parsedAge}");
            }
        }
        catch (FormatException)
        {
            Console.WriteLine("Format Exception Caught");
        }
        catch (ArgumentNullException)
        {
        }
    }
}

```



```

        Console.WriteLine("ArgumentNull Exception Caught");
    }
    finally
    {
        Console.WriteLine("Finally block executed");
    }
}

```

23.

```

using System;

class Program
{
    static string txtAge;
    static DateTime selectedDate;
    static int parsedAge;

    static void Main(string[] args)
    {
        try
        {
            Console.WriteLine(txtAge == null ? "txtAge is null" : txtAge);

            Console.WriteLine(selectedDate == default(DateTime)
                ? "selectedDate is default"
                : selectedDate.ToString());

            if (string.IsNullOrEmpty(txtAge))
            {
                Console.WriteLine("txtAge is null or empty, cannot parse");
            }
            else
            {
                parsedAge = int.Parse(txtAge);
                Console.WriteLine($"Parsed Age: {parsedAge}");
            }
        }
        catch (FormatException)
        {
            Console.WriteLine("Format Exception Caught");
        }
        catch (ArgumentNullException)
        {
            Console.WriteLine("ArgumentNullException Caught");
        }
        finally
        {
            Console.WriteLine("Finally block executed");
        }
    }
}

```

24.

```

using System;

delegate void Operation();

class Program
{
    static void Main()
    {
        Operation ops = null;

        ops += Step1;
        ops += Step2;
        ops += Step3;

        try
        {
            ops();
        }
        catch (Exception ex)
        {
            Console.WriteLine("Caught: " + ex.Message);
        }
    }
}

```

```

        Console.WriteLine("End of Main");
    }

    static void Step1()
    {
        Console.WriteLine("Step 1");
    }

    static void Step2()
    {
        Console.WriteLine("Step 2");
        throw new InvalidOperationException("Step 2 failed!");
    }

    static void Step3()
    {
        Console.WriteLine("Step 3");
    }
}

```

25.

```

using System;

namespace MethodOverloadingExample
{
    class Program
    {
        static void Main(string[] args)
        {
            int x = 5;
            new Base().F(x);
            new Derived().F(x);

            Console.ReadKey();
        }
    }

    class Base
    {
        public void F(int x)
        {
            Console.WriteLine("Base.F(int)");
        }
    }

    class Derived : Base
    {
        public void F(double x)
        {
            Console.WriteLine("Derived.F(double)");
        }
    }
}

```

26.

```

using System;

class StepEventArgs : EventArgs
{
    public int Step { get; }
    public StepEventArgs(int s) => Step = s;
}

class Workflow
{
    public event EventHandler<StepEventArgs> StepStarted;
    public event EventHandler<StepEventArgs> StepCompleted;

    public void Run()
    {
        for (int i = 1; i <= 3; i++)
        {
            StepStarted?.Invoke(this, new StepEventArgs(i));
            Console.WriteLine($"[{i}]");
            StepCompleted?.Invoke(this, new StepEventArgs(i));
        }
    }
}

```

```

}

class Program
{
    static void Main()
    {
        Workflow wf = new Workflow();

        wf.StepStarted += (s, e) =>
        {
            Console.WriteLine("<S" + e.Step + ">");
            if (e.Step == 2)
                ((Workflow)s).StepCompleted += (snd, ev)
                    => Console.WriteLine("Dyn" + ev.Step + "");
        };
        wf.StepCompleted += (s, e) => Console.WriteLine("<C" + e.Step + ">");

        wf.Run();
    }
}

```

27.

```

using System;
public class Program
{
    public static void Main(string[] args)
    {
        int a=1;
        Console.WriteLine(++a.ToString()+" "+(a++).ToString()+" "+(a++).ToString());
    }
}

```

28.

```

using System;
public class Program
{
    public static void Main(string[] args)
    {
        int a=new int();
        Console.WriteLine((a=a++).ToString()+" "+(a++).ToString()+" "+(a++).ToString());
    }
}

```

29.

```

using System;
Console.WriteLine("Hello, C#!");
int name=7;
Console.WriteLine($"Namespace: {name}");
throw new Exception();
int z;

```

30.

```

using System;
Console.WriteLine("Hello, C#!");
int name=new int()-1;
Console.WriteLine($"Namespace: {name++/name}");

```

31.

```

using System;
public class Program
{
    static void Main()
    {
        try
        {
            int i=int.MaxValue;
            Console.WriteLine(-(i+1)-i);
            for(i=0; i<=int.MaxValue;i++); //Please note the semicolon here!
            Console.WriteLine("Program ended!");
        }
        catch(Exception ex)
        {
            Console.WriteLine(ex.ToString());
        }
    }
}

```

32.

```

using System;
partial class MyLambda
{
    private bool f1;
    static void Main() => Console.WriteLine(new MyLambda(new int()).f2.ToString());
}
partial class MyLambda
{
    public int f2;
    MyLambda(int x) => Console.WriteLine(f1==(f2==0?false:true));
    MyLambda() => Console.WriteLine(f2);
}

```

33.

```

using System;
Console.WriteLine("int x = 3;");
Console.WriteLine("int y = 2 + ++x;");

int x = 3;
int y = 2 + ++x;
Console.WriteLine($"x = {x} and y = {y}");

Console.WriteLine("x = 3 << 2;");
Console.WriteLine("y = 10 >> 1;");

x = 3 << 2;
y = 10 >> 1; ;
Console.WriteLine($"x = {x} and y = {y}");

x = ~x;
y = ~y;
Console.WriteLine($"x = {x} and y = {y}");

```

34.

```

using System;
public class Program
{
    int x;
    public static void Main(string[] args)
    {
        ArrayList L=new ArrayList();
        L.Add(new Program());
        L.Add(new Program());
        for (int i=0;i<L.Count;i++)
            Console.WriteLine(++((Program)L[i]).x);

        L[0]=L[1];
        ((Program)L[0]).x = 202;

        for (int i=0;i<L.Count;i++)
            Console.WriteLine(((Program)L[i]).x);

        ((Program)L[0]).x = 111;
        L.RemoveAt(0);
        Console.WriteLine(L.Count);
        Console.WriteLine(((Program)L[0]).x);
    }
}

```

35.

```

using System;
int a = 3;
int b = a++;
Console.WriteLine($"a is {a++}, b is {-++b}");

int c = 3;
int d = ++c;
Console.WriteLine($"c is {-c--}, d is {~d}");

```

36.

```

using System;
public class Program
{
    public static void Main(string[] args)
    {
        int myInt = Convert.ToInt32("3");
        Console.WriteLine(~myInt);
    }
}

```

37.

```

        Console.WriteLine(~myInt==4?!true:false);
        Console.WriteLine("Integer less than 10? {0}", !(myInt < 10));
        Console.WriteLine("Integer between 0 and 5? {0}",!(0 <= myInt) && !(myInt <= 5));
        Console.WriteLine("Bitwise AND of Integer and 10 = {0}", myInt & 10);
    }
}

```

38.

```

using System;
public class Program
{
    public delegate void mydel();
    public void fun1()
    {
        Console.WriteLine("fun1()");
    }
    public void fun2()
    {
        Console.WriteLine("fun2()");
    }
    public static void Main(string[] args)
    {
        Program p = new Program();

        mydel obj1 = new mydel(p.fun1);
        obj1 += new mydel(p.fun2);
        Console.WriteLine("run 1");
        obj1();

        mydel obj2 = new mydel(p.fun2);
        obj2 += new mydel(p.fun1);
        Console.WriteLine("run 2");
        obj2();

        obj2 -= p.fun2;
        Console.WriteLine("run 3");
        obj2();
    }
}

```

39.

```

using System;
class Program
{
    int age;
    Program() => age=age==0?age+1:age-1;
    static void Main()
    {
        int k = "010%".Replace('0','%').Length;
        Console.Write("[ " + (k<<+new Program().age).ToString() + " ]");
        Console.Write("[ " + "010%".Split('1')[1][0] + " ]");
        Console.Write("[ " + "010%".Split('0')[1][0] + " ]");
        Console.Write("[ " + int.Parse(Convert.ToString("123".ToCharArray()[~-1])) + " ]");
    }
}

```

```

using System;
class Program
{
    int f;
    public static void Main(string[] args)
    {
        Console.WriteLine("run 1");
        Program p = new Program(new int()+"0".Length);
        for (int i = 0, _ = i; i < 5 && ++p.f >= 0; i++, Console.WriteLine(p.f++));
        {
            for (;p.f == 0;;)
            {
                Console.WriteLine(p.f);
            }
        }

        Console.WriteLine("\nrn 2");
        p = new Program(p.f);
        Console.WriteLine(p.f);
    }
}

```

```

        Console.WriteLine("\nr\n 3");
        p = new Program();
        Console.WriteLine(p.f);
    }
    Program() => f = 0;
    Program(int x) => f=x;
}

```

40.

```

using System;
class Program
{
    public static void Main(string[] args)
    {
        int[,] array2DInitialization ={{1, 2}, {3, 4}, {5, 6}, {7, 8}};
        foreach (int i in array2DInitialization)
        {
            Console.WriteLine(i);
        }
    }
}

```

41.

```

class Program
{
    Program()
    {
        Console.WriteLine("Inside constructor");
    }
    ~Program()
    {
        Console.WriteLine("Inside destructor");
    }
    public void fun()
    {
        Program obj1 = new Program();
        obj1 = null;
    }
    public static void Main(string[] args)
    {
        Console.WriteLine("Inside Main");
        Program program = new Program();
        program.fun();
        GC.Collect();
        GC.WaitForPendingFinalizers();
    }
}

```

42.

```

class Program
{
    static int i=0;
    public event funcPtr handler;
    public delegate void funcPtr();
    public void destroy()
    {
        if (i == 6)
            return;
        else
        {
            Console.WriteLine(i++);
            destroy();
        }
    }
    public static void Main(string[] args)
    {
        Program p = new Program();
        p.handler += new funcPtr((new A()).f1);
        p.handler += new funcPtr((new B()).f1);
        p.handler();
        p.handler -= new funcPtr((new B()).f1);
        p.handler -= new funcPtr((new A()).f1);
        p?.destroy();
        p = null;
        i = -6;
    }
}

```

43.

```
p?.destroy();
(new Program())?.destroy();
}
}
```

44.

```
class Program
{
    static void f(ref int x) => x--;
    static void g(int x=99) => x--;
    public static void Main(string[] args)
    {
        int x = new int();
        ref int rx = ref x;
        ref int rxx = ref rx;
        f(ref x);
        Console.WriteLine(x);
        x = 8;
        g(x);
        Console.WriteLine(x);
        f(ref rxx);
        Console.WriteLine(x);
        f(ref rx);
        Console.WriteLine(x);
    }
}
```

45.

```
class Program
{
    public static void Main(string[] args)
    {
        int x = int.Parse(Console.ReadLine());
        //int x = Console.Read();
        Console.WriteLine(x);
        switch(x%2)
        {
            case 0: { Console.WriteLine("even"); }
                    //break;
            case 1: Console.WriteLine("");
                    break;
            default: Console.WriteLine("odd");
                    break;
        }
    }
}
```

```
class Program
{
    static void f(ref int x)
    {
        x++;
    }
    static void g(int y)
    {
        y = -1;
    }
    public static void Main(string[] args)
    {
        int x = -1;
        int y = 6;
        ref int rx= ref x;
        ref int rrx = ref rx;

        //Console.WriteLine($"(Before): x={x.ToString()}, rx={rx.ToString()}, y={y.ToString()}");
        f(ref rrx);
        g(y);
        rx = 5;
        Console.WriteLine($"(After): x={x.ToString()}, rrx={rrx.ToString()}, y={y.ToString()}");
    }
}
```