1. Given 2 classes:

public class SomePublisher

{

public event EventHandler SomeEvent;

}

public class SomeSubscriber

{

public static int Count;

public SomeSubscriber(SomePublisher publisher)

{

publisher.SomeEvent += new EventHandler(publisher\_SomeEvent);

}

~SomeSubscriber()

{

SomeSubscriber.Count++;

}

private void publisher\_SomeEvent(object sender, EventArgs e)

{

// TODO: something

}

}

What would be printed after executing the following and why:

SomePublisher publisher = new SomePublisher();

for (int i = 0; i < 10; i++)

{

SomeSubscriber subscriber = new SomeSubscriber(publisher);

subscriber = null;

}

GC.Collect();

GC.WaitForPendingFinalizers();

Console.WriteLine(SomeSubscriber.Count.ToString());

**Answer: 1. Printed 0 when run in the debug mode in VS**

**2. Printed 10 when run in the release mode in VS**

**The issue is - when you use a debugger or compile wothout optimizations, the publisherobject doesn't get collected. There are  circular references in the code. The subscriber objects reference the publisher object. And the publisher object has references to the subscriber objects. The circular references prevents the garbage collector from cleaning up the publisher object, so the class destructor never gets called. Even though the subscriber object is set to Null, the class destructor doesn't get called because the publisher holds a reference to it.**

**2.** Implement a coin jar in C#. The coin jar will only accept US coinage and has a volume of 32 fluid ounces. Additionally, the jar has a counter to keep track of the total amount of money collected and has the ability to reset the count back to $0.00.