QUESTION NO: 1
You are creating a .NET Remoting object. You want to add code to the object to log error messages and warning messages. You want the log messages written to both a log file and to the Windows application log.

Which code segment should you use?

A.
```
EventLog eventLog = new EventLog(“testkobj”);
FileStream fileLog = File.Create(“testkobj.log”;
Trace.WriteLine(eventLog, “sample message”);
Trace.WriteLine(fileLog, “sample message”);
```
B.
```
EventLog eventLog = new EventLog(“testkobj”);
FileStream fileLog = File.Create(“testkobj.log”);
Trace.Write(eventLog);
Trace.Write(fileLog);
Trace.WriteLine(“sample message”);
```
C.
```
Trace.Listeners.Add(new
EventLogTraceListener(“testkobj”));
Trace.Listeners.Add(
    new TextFileTraceListener(“testkobj.log”));
Trace.WriteLine(“sample message”);
```
D.
```
Trace.Listeners.Add(new EventLogTraceListener());
Trace.Listeners.Add(
    new.TextFileTraceListener(“testkobj.log”));
Trace.WriteLine(“sample message”);
```

Answer: C
Explanation: Listeners direct the tracing output to an appropriate target, such as a log, window, or text file.
An EventLogTraceListener redirects output to an event log. A TextWriterTraceListener redirects output to an instance of the TextWriter class.
We should take care to use the **new EventLogTraceListener(“remobj”)** constructor.

Note: Any listener in the Listeners collection gets the same messages from the trace output methods. If we set up two listeners: a TextWriterTraceListener and an EventLogTraceListener. Each listener receives the same message. The TextWriterTraceListener would direct its output to a stream, and the EventLogTraceListener would direct its output to an event log.

Reference: Visual Basic and Visual C# Concepts, Trace Listeners .NET Framework Class Library, EventLogTraceListener Class [C#]
Incorrect Answers

A: The EventLog object provides interaction with Windows event logs and filestreams enables writing of data to files. However, they are not appropriate for logging warning and error messages.

B: The following statements are incorrect.
Trace.Write(eventLog);
Trace.Write(fileLog);
The correct usage is Trace.Write(Parameter), where Parameter is either an Object or a String that should be written.

D: The EventLogTraceListener Constructor() (with no parameter) initializes a new instance of the EventLogTraceListener class without a trace listener.

QUESTION NO: 2
You create a serviced component named SessionDispenser. This computer is in the TestKing.Utilities assembly and is registered in a COM+ server application. SessionDispenser has multiple callers.

You discover that there are logic problems in the Create New Session method. You want to debug any calls to this method.

What should you do?

A. Open the SessionDispenser solution.
Set a breakpoint on the CreateNewSession method.
Start the debugger.
B. Attach the debugger to the client process.
Set a breakpoint on the SessionDispenser.CreateNewSession method.
C. Attach the debugger to the TestKing.Utilites.exe process.
Set a breakpoint on the CreateNewSession method.
D. Attach the debugger to a Dllhost.exe process.
Set a breakpoint on the CreateNewSession method.

Answer: D
Explanation: Since this is a COM+ SERVER application we have to attach the debugger to the Dllhost.exe.

Reference: .NET Framework Developer's Guide, Using Serviced Components with the Global Assembly Cache

Incorrect Answers
A: The debugger must be attached to the program that should be debugged.
B: The debugger should be attached to Dllhost.exe, not to the client process.
C: We are not debugging a Library application, so we should not attach the debugger to the TestKing.Utilities.exe process.

QUESTION NO: 3
You create an XML Web service named LatLong that converts street addresses to latitude and longitude coordinates. TestKing Inc. charges for this service and allows only existing customers to use the service.

If a customer ID is not passed as part of a SOAP header, you want the service to refuse the request. You want these service refusal messages to be logged to an event log named LatLongLog. You anticipate that there will be a lot of these log entries over time. A string object named refusalMessage contains the message to log.

Which code segment should you use?

A.
```
Event log = new EventLog(“LatLongLog”);
log.WriteEntry(refusalMessage, EventLogEntryType.Error);
```
B.
```
EventLog log = new EventLog();
log.Source = “LatLongLog”; log.WriteEntry(refusalMessage,
EventLogEntryType.Error);
```
C.
```
if (!EventLog.SourceExists(“LatLongSource”))
    {   EventLog.CreateEventSource(“LatLongSource”,
        “LatLongLog”);
}   EventLog.WriteEntry(“LatLongSource”,

    refusalMessage, EventLogEntryType.Error);
```
D.
```
if (!EventLog.SourceExists(“LatLongSource”)) {
    EventLog.CreateEventSource(“LatLongSource”,
        “LatLongLog”;
}
EventLog log = new EventLog(“LatLongLog”);
log.WriteEntry(refusalMessage, EventLogEntryType.Error);
```

Answer: C
Explanation: First we use the SourcesExists method to search the registry for an existing event source. If it does not exists we create a new.

Note: The EventLog.CreateEventSource method establishes an application, using the specified Source, as a valid event source for writing entries to a log on the local computer. This method can also create a new custom log on the local computer.

Reference: .NET Framework Class Library, EventLog Members

Incorrect Answers
A, B, D: We should only create a new event source only if it does not exist.

QUESTION NO: 4

You create a serviced component named TestKingOrderProcessor. OrderProcessor implements the IOrderinit interface. The component and the interface contain the following code segments:

```
[Guid("0B6ABB29-43D6-40a6-B5F2-83A457D062AC")]
[InterfaceType(ComInterfaceType.InterfaceIsDual)]
public interface IOrderInit {
    // IOrderInit methods go here.
}

public class OrderProcessor: ServicedComponent, IOrderInit { //
    OrderProcessor methods go here.
}
```

You discover that every time you rebuild TestKingOrderProcessor, existing unmanaged client code fails. The HRESULT for the exception is 0x80040154. The exception includes the following message: "Class not registered". You need to resolve this problem.

What should you do?

A. Add a Guid attribute to the TestKingOrderProcessor class.
B. Add a ComImport attribute to the IOrderInit interface.
C. To the TestKingOrderProcessor class, add the following attribute:
   [ClassInterface(ClassInterfaceType.AutoDual)]
D. To the end of every method, add the following line of code:
   Marshal.ReleaseComObject(this);

Answer: A
Explanation: You can identify an existing COM+ target application by name or GUID. We can register the TestKingOrderProcessor class by adding a GUID attribute to it.

Reference: .NET Framework Developer's Guide, Registering Serviced Components [C#]

Incorrect Answers
B: When placed on a class, the ComImport attribute marks the class as an externally implemented Com class. However, the class must also be decorated with the Guid attribute, which specifies the CLSID for the COM class being imported.
C: The ClassInterfaceType.AutoDual method indicates that the class only supports late binding for COM clients.
D: The Marshal.ReleaseComObject method decrements the reference count of the supplied runtime callable wrapper (RCW). It is not of any use in this scenario.

QUESTION NO: 5
You create an XML Web service named PostalCode. Your project source includes a code-behind file and a file named PostalCode.asmx.

During implementation, you use the Debug class to record debugging log messages, to verify values, and to report debugging failures.

You want to deploy PostalCode to a production computer. You do not want any of the debugging code to execute on the production computer.

What should you do?

A. Set the project's active configuration to **Release** and rebuild the DLL.
B. Modify the trace element of the Web.config file by setting the enabled attribute to **"false"**.
C. Modify the compilation element of the Web.config file by setting the debug attribute to **"false"**.
D. Add code to the constructor of the PostalCode class to set the AutoFlash property of the Debug class to **false**.
E. Add code to the constructor of the PostalCode class to call the Clear method of the Debug.Listeners property.

Answer: A
Explanation: We can only exclude the debugging code from being executed by setting the Project Active Configuration to 'Release' and rebuild the Web Server again.

Note: Project build configurations are listed in the Project Property Pages dialog box and list all of the available types of project builds, such as Debug or Release

Reference: Visual Studio, Creating Solution and Project Build Configurations

QUESTION NO: 6
You create an XML Web service named TimeTKService. Each time TimeTKService is started, it checks for the existence of an event log named TimeTKServiceLog. If TimeServiceLog does not exist, TimeTKService creates it.

You discover that when TimeTKService creates TimeTKServiceLog, it throws a System.Security.SecurityException. The exception includes the following message: "Requested registry access is not allowed". You need to resolve this problem.

What should you do?

A. Configure Inetinfo.exe to run as the local administrator user account.
B. Create an installer for TimeTKService, and create the new event log in the installer code.
C. Modify the Web.config file by adding an identity element to impersonate the LOGON user specified by Internet Information Services (IIS).

D. Modify the permissions of the HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog registry key to give full control to the IUSR_*computername* user account.

Answer: A
Explanation: ASP.NET applications run under inetinfo.exe (the IIS process) or the ASP worker process aspnet_wp.exe, depending on security settings. Running as the local administrator account the IIS process would be able to create TimeServiceLog.

Reference: Visual Studio, Error: Unable to Start Debugging on the Web Server

Incorrect Answers
B: This would not allow registry access.
C: This does not work.
D: For anonymous access the IUSR_*computername* user account is used.

QUESTION NO: 7
You are creating an XML Web service named TestKingCustomer that provides customer information. You write code to keep track of error messages, warning messages, and informational messages while the service is running. You use the Trace class to write the messages to a log file.

On test computers, you want to see error messages and warning messages. On deployment computers, you want to see error messages, but not warning messages.

Which two code segments should you use? (Each correct answer presents part of the solution. Choose two)

A.
```
private static TraceSwitch mySwitch;
static BankCustomer {
    mySwitch = new TraceSwitch(“tswitch”, “a
        trace switch”);
}
```
B.
```
public static TraceLevel level;
static BankCustomer {
    level = TraceLevel.Error;
}
```
C.
```
Trace.WriteLineIf(mySwitch.TraceError,
    “An error occurred.”);
Trace.WriteLineIf(mySwitch.TraceWarning,
    “Warning message”);
```
D.
```
Trace.WriteLineIf(level == TraceLevel.Error,
    “The operation succeeded.”);
Trace.WriteLineIf(level == TraceLevel.Warning,
    “Warning message”);
```

E. Trace.WriteLineIf(mySwitch != null,
   "An error occurred.");
   Trace.WriteLineIf(mySwitch != null,
   "Warning Message");
F. Trace.WriteIf(level != TraceLevel.Off,
   "An error occurred.");
   Trace.WriteIf(level != TraceLevel.Off,
   "Warning message");

Answer: A, C
Explanation: Trace switches allow you to enable, disable, and filter tracing output. Typically, a deployed application is executed with its switches disabled.
A: To use a switch you must first create a switch object
C: We then use the WriteLineIf statement and test the switch to see if we should trace errors or warnings.

Reference: Visual Basic and Visual C# Concepts, Trace Switches .NET Framework Class Library, Trace.WriteLineIf Method [C#]

QUESTION NO: 8
You create a serviced component named TestKScheduler. TestKScheduler is registered in a library application. The Scheduler methods parse String objects into Date Time objects.

You write a console application named Coverage.exe to test each method in Scheduler. You want Coverage.exe to test Scheduler for multiple cultures to verify its globalization support.

What should you do?

A. Create a CultureInfo object for each culture locale before calling the TestKScheduler methods.
B. Create a RegionInfo object for each culture locale before calling the TestKScheduler methods.
C. Set the current thread's CurrentCulture property to each culture locale before calling the TestKScheduler methods.
D. Create a Coverage.exe.config file and add a <location> element to the configuration file for each culture locale.

Answer: C
Explanation: We set the CurrentCulture property to a local culture, then we call the TestKScheduler method. We repeat this step for each local culture.

Reference: Visual Studio, Globalization Testing

Incorrect Answers
A: CultureInfo objects would not by themselves be tested.
B: RegionInfo objects would not by themselves be tested.
D: This is not how to set up this.

QUESTION NO: 9
You have an ASP.NET application named TKWebApp. This application uses a private assembly named Employee to store and retrieve employee data. Employee is located in the bin directory of TKWebApp.

You develop a new ASP.NET application named TKWebApp2 that also needs to use Employee. You assign Employee a strong name, set its version to 1.0.0.0, and install it in the global assembly cache. You then create a publisher policy assembly for version 1.0.0.0 and install it in the global assembly cache.

You compile TKWebApp2 against version 1.0.0.0. You do not recompile MyWebApp. You then run TKWebApp.

What is the most likely result?

A. A VersionNotFoundException is thrown.
B. Employee is loaded from the bin directory.
C. Version 1.0.0.0 of Employee is loaded from the global assembly cache. D.
Version 1.0.0.0 of Employee is loaded by the publisher policy assembly.

Answer: D
Explanation: Vendors of assemblies can state that applications should use a newer version of an assembly by including a publisher policy file with the upgraded assembly.

Reference:
.NET Framework Developer's Guide. Creating a Publisher Policy
File .NET Framework Developer's Guide, Versioning

Incorrect Answers
A: A VersionNotFoundExceptio represents the exception that is thrown when attempting to return a version of a DataRow that has been deleted. **B,**
C: The Publisher Policy Assembly will be used.

QUESTION NO: 10
You are creating a Windows-based application named TKWinApp. To the application, you add a Windows Form named TKForm and a reference to a SingleCall .NET Remoting object named TheirObject.

You need to ensure that TKForm creates an instance of TheirObject to make the necessary remote object calls.

Which code segment should you use?

A. RemotingConfiguration.RegisterActivatedClientType(
        typeof(TheirObject),
        "http://TestKingServer/TheirAppPath/TheirObject.rem");
    TheirObject theirObject = new TheirObject();

B. RemotingConfiguration.RegisterWellKnownClientType(
        typeof(TestKingObject);
        "http://TestKingServer/TheirAppPath/TheirObject.
    TheirObject theirObject = new TheirObject();

C. RemotingConfiguration.RegisterActivatedServiceType(
        typeof(TheirObject));
      TheirObject theirObject = new TheirObject();

D. RemotingConfiguration.RegisterWellKnownServiceType(
        typeof(TheirObject),
        "http://TestKingServer/TheirAppPath/TheirObject.rem",
        WellKnownObjectMode.Singleton);
    TheirObject theirObject = new TheirObject();

Answer: B
Explanation: The RemotingConfiguration Class provides various static methods for configuring the remoting infrastructure. The **RegisterWellKnownClientType** method registers an object Type on the client end as a well-known type (single call or singleton).

Reference: .NET Framework Class Library, RemotingConfiguration Members

Incorrect Answers
A: The **RegisterActivatedClientType** method registers an object Type on the client end as a type that can be activated on the server.
C: The **RegisterActivatedServiceType** method registers an object Type on the service end as one that can be activated on request from a client.
D: The **RegisterWellKnownServiceType** method registers an object type on the service end as a well-known type (single call or singleton).