Question Bank –MS .NET (Solved)

I: Match the pairs

A

			.NET Layers
1	ASP.NET	a	CLR
2	ADO.NET	b	Top Layer
3	Memory Management	c	Middle Layer

Answer: 1 = b; 2 = c; 3 = a

B

			.NET Layers
4	Windows Forms	a	Middle Layer
5	Threading	b	CLR
6	Lifecycle Monitoring	c	Top Layer

Answer: 4 = c; 5 = a; 6 = b

C

	Keywords in C#		
7	unsafe	a	no more inheritance
8	sealed	b	used to alias type
9	using	c	use pointers in C#

Answer: 7 = c; 8 = a; 9 = b

D

10	C++	a	byte code
11	C#	b	no virtual machines
12	JVM	c	multiple inheritance
13	CLR	d	native code
14	JIT	e	System namespace
15	Java	f	Platform independent

Answer: 10 = c; 11 = e; 12 = a; 13 = b; 14 = d; 15 = f

E

16	CLR	a	Finalize ()
17	Garbage Collector	b	Cross language integration
18	Threading	c	System
19	Namespace	d	ASP.NET
20	Web forms	e	.NET framework

Answer: 16 = b; 17 = a; 18 = e; 19 = c; 20 = d

F

21	HashTable	a	System.Object
22	GetHashCode	b	System.Collections
23	StackTrack	c	System.String
24	PadRight	d	System.Exception
25	DoEvents	e	System.Winforms.Application

Answer: 21 = b; 22 = a; 23 = d; 24 = c; 25 = e

G

26	PadLeft	a	System.Drawing.Graphics
27	DrawLine	b	System.Winforms.Application
28	MemberwiseClone	c	System.Collections
29	TargetSite	d	System.Object
30	Queue	e	System.Exception

Answer: 26 = b; 27 = a; 28 = d; 29 = e; 30 = c

H

31	SQLConnection	a	System.IO
32	Debug	b	System.Reflection
33	Path	c	System.Data
34	Log	d	System.Security
35	Assembly	e	System.Math
36	Policy	f	System.Diagnostics

Answer: 31 = c; 32 = f; 33 = a; 34 = e; 35 = b; 36 = d

I

37	ADOConnection	a	System.IO
38	Trace	b	System.Reflection
39	StreamWriter	c	System.Data
40	Min	d	System.Security
41	Module	e	System.Math
42	Cryptography	f	System.Diagnostics

Answer: 37 = c; 38 = f; 39 = a; 40 = e; 41 = b; 42 = d

J

43	PInvoke	a	VB.NET
44	Long	b	referring Win 32 API
45	GoSub	c	ASP.NET
46	Web services	d	VB 6.0
47	WFC	e	C#
48	sbyte	f	J++

Answer: 43 = b; 44 = a; 45 = d; 46 = c; 47 = f; 48 = e

K

49	CLR	Common Language Runtime
50	CLS	Common Language Specification
51	WFC	Windows Foundation Classes
52	CTS	Common Type System
53	MSIL	Microsoft Intermediate Language
54	GC	Garbage Collector

II: State true or false

1 CLR consist of only one JIT compiler (False)
2 The JIT compilers converts the source code to the MSIL (False)
3 XML, threading forms the middle layer of .NET (True)
4 CLR is the environment in which we run .NET applications that have been compiled to MSIL (True)
5 The JIT compilers convert the MSIL to the native code (True)
6 .NET programs are interpreted by a virtual machine (False)
7 Visual Studio.NET does not have Visual Interdev (True)
8 The CLR is limited to Microsoft platform (False)
9 For .NET, we need not need the registry (True)
10 We can not use pointers in C# (False)
11 C# does not support the destructor function (True)
12 C# allows overloading assignment operator (=) (False)
13 The use of “break” in the switch-case statements in C# is compulsory (True)
14 C# is not platform independent (True)
15 C# is strongly typed (True)
16 In C#, the conversion between value and reference types are not allowed (False)
17 All the primitive datatypes in C# are objects with methods (True)
18 We can use global functions in C# (False)
19 The default value for any “int” variable in C# is zero (False)
20 C# does not have “foreach” loop (False)
21 C# does not support multiple inheritance (True)
22 A C# interface is same as a C++ abstract class (False)
23 C# destructors are same as C++ destructors (False)
24 C# do array bounds checking (True)
25 C# object-oriented language (True)
26 C# have its own class library (False)
27 All methods are virtual in C# (False)
28 C# have a 'throws' clause (False)
29 VB.NET does not support shared members (False)
30 VB.NET supports inheritance (True)
31 All VB.NET functions can be overridden (False)
32 VB.NET does not support overriding of functions (False)
33 Default methods and properties are not supported in VB.NET (True)
34 VB.NET does not have ‘variant’ data type (True)
35 VB.NET supports set and get (False)
36 Parameters in VB.NET are ByVal by default (True)
37 In VB.NET, the ‘Type’ statement is no longer available (True)
38 We can write parameterized constructor in VB.NET (True)
39 VB.NET supports multiple inheritance (True)
40 “MustInherit” keyword used with a class in VB.NET is similar to the abstract class in C++ (True)

41 The “Currency” datatype of VB 6.0 is also supported by VB.NET (False)
42 Fixed length strings are no more supported by VB.NET (True)
43 VB.NET supports “Get” (True)
44 VB.NET still supports the use of DAO, RDO and older ADO object models for accessing data (True)
45 VB.NET supports data binding to controls with DAO or RDO (False)
46 The code migration tool in VB.NET will not fix logic that uses -1 explicitly for “True” (True)
47 Ado can be used for data binding in VB.NET (True)
48 VB.NET implements exception handling through the use of Try, Catch and Finally code blocks (True)
49 The type System.sbyte from System namespace of .NET framework has a VB.NET alias sByte (False)
50 The type System.UInt16 from System namespace of .NET framework has a VB.NET alias sByte (False)
51 Not all the types from System namespace of .NET framework has VB.NET alias (True)
52 All the types from System namespace of .NET framework has C# alias (True)

<u>III: Fill in blanks</u>

1 The ___ layer of .NET includes user and program interface (top)
2 In ___, we run .NET applications that have been compiled to MSIL (CLR)
3 The ___ is used to convert MSIL to native code in .NET (JIT compilers)
4 ___ means data about data (Metadata)
5 Every type supported by Common Type System is derived from ___ (System.Object)
6 The ___ method of System.Object is used to get a Type object that can be used to access metadata (Type GetType())
7 The ___ method of System.Object is used to test equality with another object (Boolean Equals(Object))
8 The ___ method of System.Object generates a number corresponding to the value of an object (Int32 GetHashCode())
9 ___ is responsible for removing objects from heap that are no longer referenced (Garbage Collector)
10 The Garbage Collector calls ___ method immediately before the object is collected (Object.Finalize)
11 The memory management in CLR is done using ___ (Garbage Collector)
12 The datatypes for C# are stored in ___ namespace (System)
13 C# source code is compiled to higher level ___ (Intermediate Language/IL/MSIL)
14 ___ keyword is used in C# so t hat the data members will be visible within the component but not to the client (Internal)
15 In C#, conversion from value type to reference type is known as ___ (boxing)
16 In C#, conversion from reference type to value type is known as ___ (unboxing)
17 In C#, to pass parameters by reference to the function, we use ___ keyword (ref)
18 In C#, to define a class that provides an interface and some properties to the child class but should never itself be instantiated, we use ___ keyword (abstract)
19 For inheritance, the base class in VB.NET uses ___ keyword (inherits)

20 In VB.NET, members of base class can be overridden in subclasses if they are declared with the keyword ___ (Overridable)

21 In VB.NET, members of base class can not be overridden by subclasses if they are declared with the keyword ___ (NotOverridable)

22 In VB.NET, ___ keyword is used in a member in a subclass to indicate that the member is overriding the member of the same name in the base class (Overrides)

23 In VB.NET, ___ is an attribute of a class declaration for a base class which indicates that the class cannot be instantiated directly, it must be inherited and then the subclass can be instantiated (MustInherit)

24 Dim strArray (10) as String
The above code in VB.NET will create ___number of elements (Ten)

25 Dim strArray (10) as String
In above code in VB.NET, the array index will Start at zero and end at ___ (Nine)

26 The user defined types are replaced by ___ in VB.NET (Structures)

27 Both DHTML pages and WebClasses have been replaced with ___ in VB.NET (Web Forms)

28 ___ keyword is used for shared members in VB.NET (Shared)

29 The ___ member in VB.NET is identical to the static member in C++ (shared)

30 In VB.NET, to launch a new thread, an object of ___ type is declared (Thread)

31 The “Short” data type in VB.NET is ___ bits size (16)

32 The “Integer” data type in VB.NET is ___ bits size (32)

33 The “Long” data type in VB.NET is ___ bits size (64)

34 The char data type in VB.NET is ___ bytes long (2)

35 ___ is base class for all fatal runtime errors in .NET framework (CoreException)

36 ___ is thrown by the runtime when an array is accessed with an invalid index in the .NET framework (IndexOutOfRangeException)

37 ___ is the base class for all Argument exceptions in .NET framework (Argument Exception)

38 ___ is thrown by methods when in an invalid state in .NET framework (InvalidOperationException)

39 ___ is thrown when a null value is specified for a required argument (ArgumentNullException)

40 ___ is thrown when an array with wrong number of dimensions is passed to a method (RankException)

41 ___ is the base class for all exceptions (Exception)

42 ___ is the base class for all runtime generated errors (SystemException)

43 ___ is thrown by runtime when a method or a property is called on a null object reference (NullReferenceException)

44 ___ is thrown when one of the arguments to a method is not in a given range (ArgumentOutOfRange)

45 ___ is thrown by the runtime when there is not enough memory to continue the execution of the program (OutOfMemoryException)

46 ___ is thrown when an invoked method is not supported (NotSupportedException)

47 ___ is thrown when an invoked method is not supported on the current platform (PlatformNotSupportedException)

48 ___ is thrown by the runtime when an internal CLR error occurs (ExecutionEngineException)

49 ___ is thrown when the format of an argument is invalid (FormatException)

50 The ___ method of System.Exception class in .NET framework sets the value that should be returned from the HelpLink property (SetHelpLink)

51 The ___ method of System.Exception class in .NET framework returns an instance of a System.Exception class that is original innermost exception in the exception chain (GetBaseException)

IV: Multiple Choices

1 The .NET framework consist of

A ADO.NET **B XML**

C Threading **D All of above**

Answer: D

2 The top layer of .NET includes

A IO **B Windows forms**

C Net **D ASP.NET**

Answer: B,D

3 JIT compilers convert MSIL into

A native code **B byte code**

C hexadecimal code **D none of above**

Answer: A

4 In .NET, garbage collection is done in

A Top layer **B .NET framework**

C CLR **D No garbage collectioin**

Answer: C

5 The following operators can be overloaded in C#

A = = **B !=**

C = **D All of above**

Answer: A, B

6 What will be the output of the following C# code for intValue = 1

```
Switch (intValue)
{
    case 1:
            Console.WriteLine (“1”);
    case 2:
            Console.WriteLine (“2”);
    default:
            Console.WriteLine (“0”);
}
```

A	**Compile time error**	**B**	**Run time error**
C	**1**	**D**	**120**

Answer: A

7 What will be the output of the following C# code

```
    using System;
    public class Conversion{
    public static int Main()
    {
        bool b;
        int i,j;
        b = true;
        i = b;
        b = false;
        j = b;
        Console.WriteLine(i);
        Console.WriteLine(j);
        return 0;
    }
}
```

A	**1, 0**	**B**	**-1, 0**
C	**Compile time error**	**D**	**Run time error**

Answer: C

8 What will be the output of the following C# code

```
    using System;
    public class Conversion{
    public static int Main()
    {
        bool b;
        int i;
        b = false;
        i = b.ToInt32();
        Console.WriteLine(i);
        return 0;
    }
}
```

A	**0**	**B**	**Compile time error**
C	**Run time error**	**D**	**None of above**

Answer: A

9 Following operators can be overloaded in C#

A	**Arithmetic operators**	**B**	**Equality operator**
C	**Inequality operator**	**D**	**All of above**

Answer: D

10 Following datatypes are available in C#

A decimal B byte

C ushort D All of above

Answer: D

11 The default type of parameters passed to function in C# is

A by value B by reference

C no default value D none of above

Answer: A

12 What will be the output of the following C# code

```
using System
public class Sample
{
    public static void ChangeValue (int y)
    {
            y = 20;
            return;
    }
public static int Main()
{
    int x= 10;
    ChangeValue (X);
    Console.WriteLine(X);
    Console.ReadLine();
    Return 0;
}
}
```

A 10 B 20

C Compile time error D Run time error

Answer: A

13 What will be the output of the following C# code

```
using System
public class Sample
{
    public static void ChangeValue (ref int y)
    {
            y = 20;
            return;
    }
public static int Main()
{
```

```
            int X= 10;
            ChangeValue (ref X);
            Console.WriteLine(X);
            Console.ReadLine();
            Return 0;
        }
    }
```

A 10 B 20

C Compile time error D Run time error

Answer: B

14 The default value for the variables in C# is

A 0 B Garbage

C No default value, variable must be initialized

D None of above

Answer: C

15 What will be the output of the following C# code

```
        using System;
        public class DefaultVariable
        {
            public static int Main()
            {
            int k;
            Console.WriteLine(k);
            K = 10;
            Console.WriteLine(k);
            Return 0;
        }
    }
```

A Compile time error B Run time error

C 0,10 D 10,10

Answer: A

16 The "Final" class in Java is similar to ___ in C#

A Virtual class B Sealed class

C Abstract class D None of above

Answer: B

17 The "Final" variable in Java is similar to ___ in C#

A constant variable B volatile variable

C Normal variable D none of above

Answer: A

18 VB.NET does not support

A if B switch case

C GoTo **D None of above**
Answer: C

19 VB.NET does not support
A Let **B Set**
C Gosub **D All of above**
Answer: D

20 VB.NET supports
A VarPtr **B ObjPtr**
C StrPtr **D None of above supported**
Answer: D

21 Following are the keywords from VB 6.0 which are not in use in VB.NET
A DefBool **B DefByte**
C DefVar **D All of above**
Answer: D

22 Following are the keywords from VB 6.0 which are not in use in VB.NET
A DefInt **B DefLng**
C DefDbl **D All of above**
Answer: D

23 Following are the keywords from VB 6.0 which are not in use in VB.NET
A DefCur **B DefSng**
C DefDec **D All of above**
Answer: D

24 Following are the keywords from VB 6.0 which are not in use in VB.NET
A DefDate **B DefObj**
C DefStr **D All of above**
Answer: D

25 The shared members in VB.NET are similar to __ in C++
A Static members **B Local members**
C Global members **D None of above**
Answer: A

26 VB.NET has following object capabilities

A Inheritance **B Multiple Inheritance**
C Shared members **D All of above**
Answer: A, B

27 What will be the result of the following VB.NET code
MsgBox (“Hello, World”)

A Compile time error B Run time error
C Displays string Hello, World
D Message box displaying Hello, World
Answer: D

28 What are the replacement for "Print" method of VB6.0 in VB.NET
A Write B WriteLine
C WriteIf D WriteLineIf
Answer: A, B, C, D

29 Changes for Debug object in VB.NET is at
A No change B System.Debug
C Diagnostics.Debug D System.Diagnostics.Debug
Answer: D

30 In VB.NET, value of "True" becomes
A -1 B 0
C 1 D No numerical value
Answer: C

31 The following C# code
sealed MyClass
indicates
A The object can not be created
B No parent to the class MyClass
C No child to the class MyClass
D None of above
Answer: C

32 The following code in VB.NET
MustInherits MyClass
is similar to
A Abstract class of C++
B Interface implementation in C#
C Both A and B
D None of A and B
Answer: A

33 Two sample code are given below
1) MsgBox "This is VB.NET"
2) MsgBox ("This is VB.NET")
Which will work in VB.NET
A Only 1 B Only 2
C Both 1 and 2 C None of 1 and 2
Answer: B

34 Two sample code are given below
1) Dim sDate as String

sDate = Date
2) Dim sDate as String
sDate = Date()
Which will work in VB.NET

A Only 1 B Only 2
C Both 1 and 2 C None of 1 and 2

Answer: B

35 In VB.NET, arrays can be declared with

A Dim B Redim and empty index
C Redim with resize array D All of above

Answer: A, C

36 Two sample code are given below
1) Dim frmNewForm as New Form2
frmNewForm.Show
2) Dim frmNewForm as New Form2 ()
frmNewForm.Show ()
Which will work in VB.NET

A Only 1 B Only 2
C Both 1 and 2 C None of 1 and 2

Answer: B

37 In VB.NET, the And, Or, Xor, Not operators can be used for

A Boolean operations B Bitwise operations
C Both A and B D These operators can not be used

Answer: A

38 Which of the following types of System namespace of .NET framework are supported by VB.NET

A System.Byte B System.sByte
C Both A and B D None of A and B

Answer: A

39 Which of the following types of System namespace of .NET framework are supported by VB.NET

A System.Int16 B System.Int32
C System.Int64 D All of above

Answer: D

40 Which of the following types of System namespace of .NET framework are supported by VB.NET

A System.UInt16 B System.UInt32
C System.UInt64 D None of above

Answer: D

41 Which of the following interfaces are provided by System.Collections namespace

A IEnumerable B IEnumerator
C IList D All of above
Answer: D

42 Which of the following are the collection classes included in System.Collections namespace
A ArrayList B BitArray
C SortedList D All of above
Answer: D

43 Which of the following are methods for Garbage Collector
A GC.Collect B GC.RequestFinalizeOnShutdown
C GC.Finalize D All of above
Answer: A, B

44 Which of the following are CLS types
A System.UInt16 B System.UInt32
C System.UInt64 D None of above
Answer: D

45 Which of the following are CLS types
A System.Int16 B System.Int32
C System.Int64 D All of above
Answer: D

46 Which of the following are CLS types
A System.Byte B System.sByte
C Both A and B D None of A and B
Answer: A

47 Which of the following are methods for System.Object
A Equals() B GetHashCode()
C ToString D All of above
Answer: D

48 Which of the following are methods or properties of System.Exception
A Message B HelpLink
C Both A and B D None of A and B
Answer: C

49 Which of the following are collection classes in System.Collections
A Queue B Stack
C Both A and B D None of A and B
Answer: C

50 Which of the following is not keyword in C#

A	**sealed**	**B**	**Overrides**
C	**internal**	**D**	**abstrac**

Answer: B

V. Short Answer Questions:

1) How can I get at the Win32 API from a .NET program?
Use PInvoke. This uses similar technology to COM Interop, but is used to access static DLL entry points instead of COM objects.

2) What is garbage collection?
Garbage collection is a mechanism that allows the computer to detect when an object can no longer be accessed. It then automatically releases the memory used by that object (as well as calling a clean-up routine, called a "finalizer," which is written by the user). Some garbage collectors, like the one used by .NET, compact memory and therefore decrease your program’s working set.

3) Are C# constructors the same as C++ constructors?
C++ programmers may be used to writting additional methods for initialzation because there are no return values available in constructors. However, in C# you can throw an exception to get back a result from the constructor.

4) Are C# destructors the same as C++ destructors?
No! They look the same but they’re very different. First of all, a C# destructor isn’t guaranteed to be called at any particular time. In fact it’s not guaranteed to be called at all. Truth be told, a C# destructor is really just a Finalize method in disguise. In particular, it is a Finalize method with a call to the base class Finalize method inserted.

5) Write a simple ’Hello, world’ C# program.

```
class CApplication
{
        public static void Main()
        {
                System.Console.Write( "Hello, new .NET world." );
        }
}
```

6) What standard types does C# supply?
C# supports a very similar range of basic types to C++, including int, long, float, double, char, string, arrays, structs and classes. However, don’t assume too much. The names may be familiar, but some of the details are different. For example, a long is 64 bits in C#, whereas in C++ the size of a long depends on the platform (typically 32 bits on a 32-bit platform, 64 bits on a 64-bit platform). Also classes and structs are almost the same in C++ - this is not true for C#.

7) What is difference between Structs in C++ and C#?

In C++, a struct and a class are pretty much the same thing. The only difference is the default visibility level (public for structs, private for classes). However, In C# structs and classes are very different. In C#, structs are value types (stored on the stack), whereas classes are reference types (stored on the heap). Also structs cannot inherit from structs or classes, though they can implement interfaces. Structs cannot have destructors.

8) Is a C# interface the same as a C++ abstract class?

No, not quite. An abstract class in C++ cannot be instantiated, but it can (and often does) contain implementation code and/or data members. A C# interface cannot contain any implementation code or data members - it is simply a group of method names & signatures. A C# interface is more like a COM interface than a C++ abstract class.

The other major difference is that a C# class can inherit from only one class (abstract or not), but can implement multiple interfaces.

9) What is a static constructor? When it is called?

A constructor for a class, rather than instances of a class. The static constructor is called when the class is loaded.

10) How to declare a pure virtual function in C#?

Use the abstract modifier on the method. The class must also be marked as abstract (naturally). Note that abstract methods cannot have an implementation (unlike pure virtual C++ methods).

11) Can we use exceptions in C#? What types of object can we throw as exceptions?

Yes, in fact exceptions are the recommended error-handling mechanism in C#. Most of the .NET framework classes use exceptions to signal errors.

Only instances of the System.Exception classes, or classes derived from System.Exception can be used to throw as exception.

12) Can we define our own exceptions? Does C# have a ’throws’ clause?

Yes, as long as you follow the rule that exceptions derive from System.Exception. More specifically, MS recommend that user-defined exceptions inherit from System.ApplicationException (which is derived from System.Exception).

Unlike Java, C# does not require (or even allow) the developer to specify the exceptions that a method can throw.

13) If we defined a destructor in C#, it never gets called. Why?

A C# destructor is really just an implementation of Finalize, and the runtime doesn’t guarantee to call Finalize methods. The semantics of a C# destructor are quite different from a C++ destructor.

14) Most of the C# basic types have the same names as C++ basic types? Are they the same? Explain.

No. A char in C# is equivalent to a wchar_t in C++. All characters (and strings, obviously) are Unicode in C#. Integral values in C# are concrete sizes, unlike C++ (where size depends on processor). For example, a C# int is 32 bits, whereas a C++ int is normally 32 bits on a 32-bit processor and 64 bits on a 64-bit processor. A C# long is 64 bits.

15) String comparisons using = = seem to be case-sensitive? Write a code in C# so that we should have a case-insensitive string comparison?
Use the String.Compare function. Its third parameter is a boolean which specifies whether case should be ignored or not.
"fred" == "Fred"   // false
System.String.Compare("fred", "Fred", true)   // true

16) Does C# support a variable number of arguments? Explain.
Yes, using the params keyword. The arguments are specified as a list of arguments of a specific type, e.g. int. For ultimate flexibility, the type can be object. The standard example of a method which uses this approach is System.Console.WriteLine().

17) Does C# do array bounds checking? If yes, how?
Yes. An IndexOutOfRange exception is used to signal an error.

18) How can you make sure your C# classes will interoperate with other .NET languages?
Make sure your C# code conforms to the Common Language Subset (CLS). To help with this, add the [assembly:CLSCompliant(true)] global attribute to your C# source files. The compiler will emit an error if you use a C# feature which is not CLS-compliant.

19) Are there dynamic arrays in C#? If so, how does it compare with the VB implementation?
No. But the runtime has an ArrayList class in the System.Collections namespace which works nicely for when you need arrays that can dynamically grow. (on C#, you can even reference it as you would an array since it has an indexer).

20) Discuss operator overloading in C#
C# supports operator overloading
Following operator can be overloaded in C#
a) Arithmetic operators   b) equality operators   c) inequality operators
Overloading of assignment operator is not allowed is C#

21) What are various layers in .NET application
.Net application provides three layes

1) CLR consisting of CTS, GC, etc
2) .NET framework or middle layer consisting of XML, ADO.NET, etc
3) Top layer consisting of web forms and windows forms

22) What the top layer of .NET application contains and where they are used in application
The top layes of .NET consist of

1) Web forms (ASP.NET) to create UI for web services
2) Windows forms to create standard win32 desktop applications

23) What the .NET framework of .NET contains
The .NET framework consist of

1) ADO.NET

2) XML
3) Threading
4) Security
5) Diagnostics
6) IO
7) Net

24) What the Common Language Runtime of .NET application contains

The CLR consist of

1) Common Type System (CTS)
2) Memory management using Garbage Collector (GC)
3) JIT compilers

25) Why Common Type System is used in .NET applications

One of the most advantage of .NET is that it is used for cross language integration.

For this there should be some mechanism which will provide simillr data types for various languages which supports .NET like C# and VB.NET

For this purpose, the CTS is used.

CTS provides common data types for various languages. All these types are derived from System.Object, hence they all can use the methods provided by System.Object

26) What a note on JIT compilers

The JIT compilers are present in the CLR. They converts the MSIL in to the native code which is understandable by machine

Following are the JIT compilers

Econo-JIT

27) .NET provides transparency of source code, explain

The bytecode in the IL are much higher level than the processor instructions that programs are compiled into today. While we disassembled a program in current environments, the assembly base results are limited use.

.NET programs disassembled from IL,o the other hand,will more closely resemble actual source code. They will also contain information to understand data structure. Thus make the code more transparent.

28) .NET is language incompatible, explain

Making language work in the new framework of .NET usually means adjustment to the language syntax. This introduces language compatibility problems in moving existing code into .NET.

For example the VB.NET is a particular problem.

29) Who manages the security in .NET applications and how

The CLR manages the security in .NET applications.

The CLR enables managed code to declare the security they required and the gathering of evidence used to evaluate whether or not permission is granted.

30) Who manages versioning and deployment support in .NET application and how

The CLR manages versioning and deployment support in .NET application.
The CLR supports side by side execution of multiple versions of the same component, even within the same process.

31) What are various methods supported by Common Type System

1) Boolean Equals (Object)
2) Int32 GetHashCode ()
3) Type GetType ()
4) String ToString ()

32) Give at least 6 primitive types supported by .NET applications

Boolean,Byte, Char, DateTime, Decimal, Double, GUID, Int16, Int32, Int64

33) Explain, why C# is safer than C++

C# prohibits certain C++ idioms that tends to cause bugs. These includes preprocessor macros, fall through switch case statements, and pointers.
Also the GC is responsible for the collection of objects which are no longer referenced, the C# programs are safe from memory leaks.

34) Give the advantages of C# over Java

1) C# supports typesafe enumerations
2) C# supports operator overloading
3) C# is interoperable

35) If you want to use a language which is platform independent, what will you use, C# or Java. Explain why

We must use Java. Java is platform independent while C# is not. C# requires .NET runtime and that runtime has not been implemented for non windows platforms, C# is platform dependent.

36) If you want to use a language which is interoperable, what will you use, C# or Java. Explain why

We must use C#.
As C# uses .NET runtime, it can interoperate with code written in other .NET programming language. With Java, this is very complicated.

37) Give at least 6 primitive datatypes supported by C#

bool, int, byte, char, double, float, sbyte, short, string, struct, ulong, ushort , uint.

38) What care must be taken by a C++ programmer for writing code in C#

1: break in switch case is compulsory
2: main is static function
3: there is no global function in C#
4: variables must be initialized
5: multiple inheritance is not possible in C#, use interface implementing

39) Write a note on Enumerations in C#

An enumeration is custom integer type. When we declare enumeration, we specify a set of values that an instance of that type can contain. In C#, we can define enumeration.
For eg,

Private enum GenderEnum (Male = 0, Female = 1);

Once declared, enums can be used to declare a data member that represent the state.
Enum makes C# code more correct by leveraging the compiler's power to flag aatempted assignments of illegal values to enumeration values.

40) What are the operators which can be overloaded in C#
Operators which can be overloaded in C# are

a) Arithmetic operators
b) Equality operators
c) Inequality operators

41) Give the 4 access modifiers for C# classes
The 4 access modifiers for C# classes are

1) private
2) public
3) protected
4) internal

42) What care should be taken while writing switch – case statements for C# and why?
While writing switch case statement for C#, we must use the 'break' keyword after every case statememt.
In C++, the use of break is not compulsory, which sometimes creating bugs, and are difficult to interpret. To avoid this, the C# compiler requires that every case statement must be terminated by break statement.

43) What are the keywords used for error handling in C#
Keywords used for handling errors in C# are
Try, Catch, Finally.

44) Give at least 3 Exception classes for C#
IndexOutOfRangeException, NullReferenceException, InvalidCastException.

45) If we want that the class should not be parent for other classes, what type of classes we must use in C#. If the code is written in Java, what class should be used?
For this, we must use sealed keyword

```
Eg--  sealed public class SealedClass
{
//code
}
```

For Java, use Final keyword with the class

46) What are advantages of Java over C#

Advantages of Java over C#

1) Java is platform independent while C# is not.
2) The beta version of C# still having lots of bugs
3) In Java the code like

```
If ( m_Var=10)
{
//code
}
```

Is not allowed which hgives wrong answer in C# and other languages.

47) What are the datatypes from VB 6.0 are not supported by VB.NET. Which new datatypes are added in VB.NET?
Datatypes not supported by VB.NET are
Varient, Currency, fixed length strings
New datatypes added to VB.NET are
Char, Long

48) Give at least 6 datatypes supported by VB.NET
Short, Integer, Long, Char, Decimal, Byte

49) In VB 6.0 and VB.NET, what is default for parameters, ByVal or ByRef?
For VB 6.0, parameters are ByRef by default.
For VB.NET, parameters are ByVal by default.

50) Which are the keywords from VB 6.0 not used in VB.NET
GoSub, GoTo Let, VarPtr, ObjPtr, StrPtr, DefBool, DefByte

51) Which are the types of System.Object, which are also CTS type
System.Byte, System.Int16, System.Int32, System.Int64

52) Which are the types of System.Object, which are not CTS type
System.sbyte, System.UInt16, System.UInt32, System.UInt64

53) Give the collection classes in the System.Collections
ArrayList, BitArray, HashTable, NameValueCollection, Queue, SortedList, Stack, StringCollection.

VI. Long Answer Questions:

1) Why doesn’t the .NET runtime offer deterministic destruction?
Because of the garbage collection algorithm. The .NET garbage collector works by periodically running through a list of all the objects that are currently being referenced by an application. All the objects that it doesn’t find during this search are ready to be destroyed and the memory reclaimed. The implication of this algorithm is that the runtime doesn’t get notified immediately when the final reference on an object goes away - it only finds out during the next sweep of the heap.

Furthermore, this type of algorithm works best by performing the garbage collection sweep as rarely as possible. Normally heap exhaustion is the trigger for a collection sweep.

2) What is the .NET Framework?

The Microsoft .NET Framework is a platform for building, deploying, and running Web Services and applications. It provides a highly productive, standards-based, multi-language environment for integrating existing investments with next-generation applications and services as well as the agility to solve the challenges of deployment and operation of Internet-scale applications. The .NET Framework consists of three main parts: the common language runtime, a hierarchical set of unified class libraries, and a componentized version of Active Server Pages called ASP .NET.

3) What is the Common Language Runtime (CLR)?

The Common Language Runtime is the execution engine for .NET Framework applications.
It provides a number of services, including the following:
Code management (loading and execution)
Application memory isolation
Verification of type safety
Conversion of IL to native code
Access to metadata (enhanced type information)
Managing memory for managed objects
Enforcement of code access security
Exception handling, including cross-language exceptions
Interoperation between managed code, COM objects, and pre-existing DLLs (unmanaged code and data)
Automation of object layout
Support for developer services (profiling, debugging, and so on)

4) What is the Common Language Specification (CLS)?

The Common Language Specification is a set of constructs and constraints that serves as a guide for library writers and compiler writers. It allows libraries to be fully usable from any language supporting the CLS, and for those languages to integrate with each other. The Common Language Specification is a subset of the common type system. The Common Language Specification is also important to application developers who are writing code that will be used by other developers. When developers design publicly accessible APIs following the rules of the CLS, those APIs are easily used from all other programming languages that target the Common Language Runtime.

5) What is managed code and managed data?

Managed code is code that is written to target the services of the Common Language Runtime (see What is the Common Language Runtime?). In order to target these services, the code must provide a minimum level of information (metadata) to the runtime. All C#, Visual Basic .NET, and JScript .NET code is managed by default. Visual Studio .NET C++ code is not managed by default, but the compiler can produce managed code by specifying a command-line switch (/CLR).

Closely related to managed code is managed data—data that is allocated and de-allocated by the Common Language Runtime's garbage collector. C#, Visual Basic, and JScript .NET data is managed by default. C# data can, however, be marked as unmanaged through the use of special keywords. Visual Studio .NET C++ data is unmanaged by default (even when using the /CLR switch), but when using Managed Extensions for C++, a class can be marked as managed by using the __gc keyword. As the name suggests, this means that the memory for instances of the class is managed by the garbage collector. In addition, the class becomes a full participating member of the .NET Framework community, with the benefits and restrictions that brings. An example of a benefit is proper interoperability with classes written in other languages (for example, a managed C++ class can inherit from a Visual Basic class). An example of a restriction is that a managed class can only inherit from one base class.

6) How does non-deterministic garbage collection affect the code?
For most programmers, having a garbage collector (and using garbage collected objects) means that you never have to worry about deallocating memory, or reference counting objects, even if you use sophisticated data structures. It does require some changes in coding style, however, if you typically deallocate system resources (file handles, locks, and so forth) in the same block of code that releases the memory for an object. With garbage collected object you should provide a method that releases the system resources deterministically (that is, under your program control) and let the garbage collector release the memory when it compacts the working set.

7) Can you avoid using the garbage-collected heap?
All languages that target the runtime allow you to allocate class objects from the garbage-collected heap. This brings benefits in terms of fast allocation, and avoids the need for programmers to work out when they should explicitly 'free' each object.

The CLR also provides what are called ValueTypes -- these are like classes, except that ValueType objects are allocated on the runtime stack (rather than the heap), and therefore reclaimed automatically when your code exits the procedure in which they are defined. This is how "structs" in C# operate.

Managed Extensions to C++ lets you choose where class objects are allocated. If declared as managed Classes, with the __gc keyword, then they are allocated from the garbage-collected heap. If they don't include the __gc keyword, they behave like regular C++ objects, allocated from the C++ heap, and freed explicitly with the "free" method.

8) Can we use COM objects from a .NET Framework program? Explain
Yes. Any COM component you have deployed today can be used from managed code, and in common cases the adaptation is totally automatic.

Specifically, COM components are accessed from the .NET Framework by use of a runtime callable wrapper (RCW). This wrapper turns the COM interfaces exposed by the COM component into .NET Framework-compatible interfaces. For OLE automation interfaces, the RCW can be generated automatically from a type library. For non-OLE automation interfaces, a developer may write a custom RCW and manually map the types exposed by the COM interface to .NET Framework-compatible types.

9) Can .NET Framework components be used from a COM program? Explain

Yes. Managed types you build today can be made accessible from COM, and in the common case the configuration is totally automatic. There are certain new features of the managed development environment that are not accessible from COM. For example, static methods and parameterized constructors cannot be used from COM. In general, it is a good idea to decide in advance who the intended user of a given type will be. If the type is to be used from COM, you may be restricted to using those features that are COM accessible.

Depending on the language used to write the managed type, it may or may not be visible by default.

Specifically, .NET Framework components are accessed from COM by using a COM callable wrapper (CCW). This is similar to an RCW (see previous question), but works in the opposite direction. Again, if the .NET Framework development tools cannot automatically generate the wrapper, or if the automatic behavior is not what you want, a custom CCW can be developed.

10) Write short note on three tire applications in .NET

.NET architecture follows 3 tire application. The 3 tire application consist of

1) Top Layer 2) .NET framework 3) CLR

The top layer consist of user and program interfaces. It consist of windows forms and web forms. Windows forms are new way to create standard win32 desktop applications. Web forms provides formed based UI for web services.

The .NET framework includes next generation of standard system services such as ADO.NET and XML.NET. These services are brought under the control of the framework, making them universally available and standardizing their usage across languages.

The CLR is the heart for the .NET applications. It includes CTS which makes cross language inheritance possible. The CLR also does memory management by garbage collection and lifecycle monitoring.

Diagram ---page 13

11) Give the importance of Common Language Runtime

The common language Runtime is the heart for the .NET applications.

The CLR includes a common system of data types. These common types and a standard interface convention, makes cross language inheritance possible.

Also the CLR does memory management. It maintains the reference counting for objects, and handles garbage collection. The garbage collection is responsible for the collection of objects which are no more referred.

CLR also does the lifecycle monitoring.

CLR also provides

- Enhanced security through managed code
- Versioning and deployment support
- Debugging and profiling services

12) What are advantages of .NET

1) Faster development

2) Lots of build in functionality through a rich object model
3) A variety of ways to interface and integrate with the outside world
4) More usable
5) Easy to integrate different languages into one system
6) Easier deployment
7) Scalability
8) Easy to build sophisticated development tools
9) Interfaces well to existing software
10) Versioning support

13) What are disadvantages of .NET

1) Language incompatibilities: Making languages work in this framework usually means adjustment to the language syntax. For example VB.NET
2) Transparency of code: The bytecode in IL are much higher level than the processor instructions that programs are compiled into today..NET programs disassembled from IL will more closely resemble the actual source code. Thus making the code more transparent.
3) Not platform independent: The languages working on .NET platform are not platform independent
4) Current beta version is not a full fledged version for .NET applications

14) How the CLR introduces the concept of managed heap

Heap is a region of one or more pages of reserved address space that can be subdivided and allocated into smaller chunks. The managed heap is a region of memory that is allocated and managed by CLR. All managed objects are allocated on the heap and the CLR is responsible for controlling access to these objects in a type safe manner. This is accomplished by only allowing managed code to directly access managed data.

Diagram—page 56

15) Write short note on System.Object

The CTS provides every language a base set of types. Every type is derived from System.Object System.Object provides following methods for every type
1)Boolean Equals(Object): Used to test equality with other object
2)Int32 GetHashCode(): It generates a number corresponding to the valus of an object
3)Type GetType(): Gets the type object that can be used to access metadata
4)String ToString(): The default implementation returns the fully qualified name of the class of the object

The types supported by System.Object are
Boolean, Byte, Char, DateTime, Decimal, Double, GUID, Int16, Int32, Int64, Sbyte, Single, TimeSpan

16) How memory management is done in .NET

The CLR is responsible for the memory management in .NET applications.

The CLR consist of garbage collector The GC is responsible for removing objects from the heap that are no longer referenced, compacting and remaining allocation ,and then finally resetting the reference to the end of the memory allocation on the heap
The CLR introduces the concept of managed heap. If the CLR is unable to allocate memory on the managed heap, the GC will be invoked which then removes objects from the heap.

17) How cross language integration is done in .NET
The .NET consist of CLR which contains CTS. The CTS provides every language running on the top of the .NET platform with a base set of types which are from System.Object. Thus all the types will support the method of System.Object which will help for cross language integration. The code written in any of he language will be compiled to the MSIL .The CLR then enables managed code written in one language to seamlessly integrate with code written in another language.

18) Write note on Garbage Collector
The GC is responsible for for removing objects from the heap that are no longer referenced, compacting and remaining allocation ,and then finally resetting the reference to the end of the memory allocation on the heap.
The GC will be invoked by the CLR when the CLR is unable to allocate memory on the managed heap. Thus the GC may be automatically be invoked by the CLR or the application may explicitly invoke the GC by calling GC.Collect.
The GC will call Object.Finalize method immediately before the object is collected. GC helps resolve many of the memory leak problems.
As long as all references to the object are either implicitly or explicitly released by the application, the GC will take care of freeing the memory allocated to the object on the heap.

19) Write note on Object.Finalize method
The Object.Finalize method is called by the GC just before the object is collected . Reference types are able to override the Finalize method to perform any necessary clean up. The Finalize is not the destructor.
The GC will usually be triggered by low memory situations. Execution of objects Finalize method will more than likely incur performance penalties due to page faults. Hence the code path for the Finalize method should be as short as possible.
All clean up activities should be placed in the Finalize method. The advantage of this is to ensure that the object will be clean up properly before the GC collects it.

20) Write short note on System namespace
The CLS describes asset of data types that a programming language must support in order to be considered as .NET compliant. These type are stored in System namespace
These system types are shared across all of the .NET languages. Languages like VB.NET and C# provides aliases for the System types
The table below gives list of types and their aliases in VB.NET and C#

Type	VB.NET	C#
System.Byte	Byte	byte
System.Int16	Short	short
System.Int32	Integer	int
System.Int64	Long	long

The types contained in System namespace are used to
1) Develop console applications and Windows services
2) Define the base data types such as Object, Integer, etc
3) Manage garbage collection
4) Raise and handle events

21) What are differences in C# and C++

1) C++ code is compiled to native machine code while C# code is compiled to MSIL
2) C# is completely object oriented language but C++ is not a complete OO language
3) C++ allows global variables which his not allowed is C#
4) C++ allows multiple inheritance which is not allowed in C#
5) The use of 'break' in switch case is not compulsory in C++ which is made compulsory in C#
6) The main is static function in C#
7) We can avoid some class from being inherited in C# which is not possible with C++

22) What are differences in C# and Java

1) Java is platform independent while C#is not platform independent
2) C# provides typesafe enumerations. Java does not provide that
3) We can overload operators in C# but not in Java
4) C# is language interoperable than Java
5) Use of break in switch case is compulsory in C# but it is not compulsory in Java

23) Give the advantages of C# over C++
1) C# is completely object oriented language
2) C# is language interoperable and C++ is not.
3) We can avoid some class from being inherited in C# which is not possible with C++
4) C# is safer than C++
5) C# is strongly typed, hence less chances of data loss

24) What are similarities in C# and Java

1) Both C# and Java are completely object oriented languages
2) Both have static main function
3) Multiple inheritance is not allowed in C# and Java, rather interface implementation is the solution.
4) No global functions in C# and Java is allowed
5) The Garbage collection is the mechanism by which objects which are no more referenced are removed in C# and Java
6) The data types are objects and stored on heap in C# and Java

25) What are advantages of C#

1) Completely object oriented language
2) Language interoperable
3) C# is safer
4) Any class can be avoided from inheriting from other class using sealed keyword
5) C# is very much similar to Java

6) Use of pointers is prohibited
7) Memory management using garbage collection

26) What are disadvantages of C#

1) C# is not platform independent
2) Slower than C++ as the code is first compiled to MSIL and then to native code
3) Only beta version is available with lots of bugs
4) Can not overload assignment operator

27) Write short note on boxing and unboxing

Boxing is a mechanism by which value type on stack is converted to the reference type on heap.
Unboxing is the mechanism by which reference type is converted to value type.
When unboxing a value, we must make sure that the type on the stack is capable ot containing the value on the heap
Following example gives the conversion of int fromstack to the heap(boxing)

```
uing System;
public class Boxing{
object objA;
int j;

int i= 10;
objA = (object)i;

j= (int) objA;
Console.WriteLine(j);
Return 0;
}
}
```

28) Write a C# code for swap function

```
using System
public class Swap
{
        public static void Swap(ref int x, ref int y)
        {
                int z;
                z= x;
                x= y
                y = z
                return;
        }
public static int Main()
{
        int x= 10;
        int y = 20
        Swap(ref x, ref y);
        Console.WriteLine(x);
        Console.WriteLine(y);
        Console.ReadLine();
        Return 0;
}
}
```

29) Explain, why Main () function in C# is static

C# is a completely object oriented language. Hence it does not allow any global functions.
In traditional c and C++ programming, the main function is global.
The main function is the entry point for the program. Hence it should be called before the object of the class is created. As it is global function in C and C++, it gets called without creating any objects. But for C#, as no global functions are allowed, the main function becomes static. The static functions of any class can be called without creating the instance of the object. Thus the static main function of the C# will be called before creating the object.
Hence the main() function in C# is static.

30) Discuss various loops in C#

Various loops in C# are

1) while loop: C# while loop is similar to C++ while loop

It is a pre test loop, in which the expression is evaluated and the statement below it is executed it the value is determined to be true, At the end, the expression is evaluated again.

Eg-

```
While (expression)
{
//code to execute until expression evaluates to false
}
```

2) do_while loop: Much like while loop, but it is executed at least once. It is a post test loop

```
do
{
//code to execute until expression evaluates to false
}while(expression)
```

3) for loop: this loop is useful for repeating a simple or compound statememt for a predetermined number of times

```
for (i = 0; i<5; i++)
{
//code to execute until i<5
}
```

4) foreach loop: It is similar to foreach loop of Java

```
foreach(int j in arrant)
        Console.WriteLine (j);
```

This loop is used to iterate through the items in an array

31) Discuss structured error handling in C#
C# provides three keywords for grouping code into blocks for handling structured errors
These are Try, Catch and Finally
The 'Try' keyword prefaces blocks of code in which errors might be generated . When an error ocures, the 'Try' block creates exception object and throws it outside the block and control leaves the try block
The catch keyword prefaces blocks of code that follows try block and catch exception objects that the try block might throw. Code inside catch will execute when erreo occurs.
The finally keyword prefaces blocks of code that must always be executed when a try block ends

32) Discuss, C# is an Object Oriented Language
C# is a completely object oriented language.
C# follows all the rules for a complete object oriented language

1) C# does not have global functions as global functions are not recommended by OOPs
2) C# supports inheritance
3) C# does not support multiple inheritance but interface implementation is supported
4) C# supports encapsulation
5) C# supports abstraction and polymorphism
6) C# supports operator overloading

Thus C# is a object oriented language.

33) Discuss various access modifiers for C# classes
Access modifiers for C# are

1) The private modifier is used when we need data members to be visible to the code within the class
2) The protected modifier is used when the data members to be visible to the code within the class and to the code in the child class
3) The internal modifier modifies the data members so that they are only visible to code within the component and not to the clients
4) The public modifier is used when the data members to be visible to all codes

34) Discuss inheritance in C#
The code reuse is object oriented idea. Implementing inheritance allows the child class to re use functions implemented in parent class from which they are inherited.
C# allows inheritance but it does not allow multiple inheritance. Instead of multiple inheritance, C# allows interface implementation.
For a abstract class, C# use 'abstract' keyword.
Eg
Abstract public class MyClass
For virtual methods, C# use virtual keyword.
Eg

```
        Public virtual System.Drawing.Color.CellColor
        {
        //code
        }
```

If we want that some class must not be inherited, use sealed keyword

Eg

```
Sealed public class SealedClass
{
//code
}
```

35) Write note on PInvoke

The Platform Invocation Services allows C# programs to invoke functions stored in C DLLs including those C DLLs that supply the Win32 APIs.

To use PInvoke, define aC# wrapper function whose signature matches that of the C function that is to be invoked and modify the wrapper function’s definition with ‘sysimport’ attribute.

To invoke the external function from C# code, call the wrapper function. The CLR load the DLL into the memory, loates the desired function in DLL, marshals the function call’s parameters and passes the thread-of-execution into the function.

Dangers of PInvoke

First, If you use PInvoke to make your C# programs directly reference the Win32 API, those programs are tied to the Windows operating system.

Second, the CLR can not apply security checks to the code in external DLLs

36) What are differences between VB.NET and VB 6.0

1) VB.NET supports inheritance while VB 6.0 does not support inheritance
2) In VB.NET, the parameters are ByVal by default, that in VB6.0 is ByRef by default
3) VB.NET does not support the set and let which VB 6.0 supports.
4) No variant data type in VB.Net
5) User defined types are no more supported by VB.NET
6) In VB.NET, bitwise operators are used for Boolean operations only. But in VB 6.0, bitwise operators are used for both Boolean and bit wise operations

37) Give the object capabilities of VB.NET

VB.NET is a complete object oriented language

1) Fully inheritance
2) Overloading of functions
3) Shared members which are similar to static members of C++
4) Parameterised constructors
5) Overriding of functions

38) If in C#, an int is a value type, and a class is a reference type. How can int be derived from object?

It isn't, really. When an int is being used as an int, it is a value (on the stack). However, when it is being used as an object, it is a reference to an integer value on the heap. In other words, when you treat an int as an object, the runtime automatically converts the int value to an object reference. This process is called boxing. The conversion involves copying the contents of the int from the stack to the heap, and creating an object instance which refers to it. Unboxing is the reverse process - the object is converted back to a stack-based value.

```
int x = 3;              // new int value 3 on the stack
object objx = x;        // new int on heap, set to value 3 - still have x=3 on stack
int y = (int)objx;      // new value 3 on stack, still got x=3 on stack and objx=3 on heap
```

39) C# uses references instead of pointers. Are C# references the same as C++ references? Expalin with code.
Not quite. The basic idea is the same, but one significant difference is that C# references can be null . So you cannot rely on a C# reference pointing to a valid object. If you try to use a null reference, a NullReferenceException is thrown.

For example, look at the following method:

```
void displayStringLength( string s )
{
        Console.WriteLine( "String is length {0}", s.Length );
}
```

The problem with this method is that it will throw a NullReferenceException if called like this:

```
string s = null;
displayStringLength( s );
```

Of course for some situations you may deem a NullReferenceException to be a perfectly acceptable outcome, but in this case it might be better to re-write the method like this:

```
void displayStringLength( string s )
{
        if( s == null )
                Console.WriteLine( "String is null" );
        else
                Console.WriteLine( "String is length {0}", s.Length );
}
```

40) In C#, how can we check the type of an object at runtime? Write a sample C# code.
Use the is keyword. For example:

```
using System;

class CApp
{
        public static void Main()
        {
                string s = "fred";
                long i = 10;

                Console.WriteLine( "{0} is {1}an integer", s, (IsInteger(s) ? "" : "not ") );
                Console.WriteLine( "{0} is {1}an integer", i, (IsInteger(i) ? "" : "not ") );
```

```
    }

    static bool IsInteger( object obj )
    {
        if( obj is int || obj is long )
            return true;
        else
            return false;
    }
}
```

produces the output:

```
fred is not an integer
10 is an integer
```

41) In C#, how can we get the name of a type at runtime? Write a sample code

Use the GetType method of the object class (which all types inherit from). For example:

```
using System;

class CTest
{
    class CApp
    {
        public static void Main()
        {
            long i = 10;
            CTest ctest = new CTest();

            DisplayTypeInfo( ctest );
            DisplayTypeInfo( i );
        }

        static void DisplayTypeInfo( object obj )
        {
            Console.WriteLine( "Type name = {0}, full type name = {1}", obj.GetType(),
obj.GetType().FullName );
        }
    }
}
```

produces the following output:

```
Type name = CTest, full type name = CTest
Type name = Int64, full type name = System.Int64
```

42) What are delegates? Write a C# code

A delegate is a class derived from System.Delegate. However the language has a special syntax for declaring delegates which means that they don't look like classes. A delegate represents a method with a particular signature. An instance of a delegate represents a method with a particular signature on a particular object (or class in the case of a static method). For example:

```
using System;
delegate void Stereotype();

class CAmerican
{
        public void BePatriotic()
        {
                Console.WriteLine( "... <gulp> ... God bless America.");
        }
}

class CBrit
{
        public void BeXenophobic()
        {
                Console.WriteLine( "Bloody foreigners ... " );
        }
}

class CApplication
{
        public static void RevealYourStereotype( Stereotype[] stereotypes )
        {
                foreach( Stereotype s in stereotypes )
                        s();
        }

        public static void Main()
        {
                CAmerican chuck = new CAmerican();
                CBrit edward = new CBrit();

                // Create our list of sterotypes.
                Stereotype[] stereotypes = new Stereotype[2];
                stereotypes[0] = new Stereotype( chuck.BePatriotic );
                stereotypes[1] = new Stereotype( edward.BeXenophobic );

                // Reveal yourselves!
                RevealYourStereotype(stereotypes );
        }
}
```

This produces the following result:

```
... <gulp>... God bless America.
Bloody foreigners ...
```

43) What is the C# equivalent of QueryInterface? Write C# code.

The ‘as’ keyword. For example:

```
using System;

interface IPerson
{
    string GetName();
}

interface IPerson2 : IPerson
{
    int GetAge();
}

class CPerson : IPerson
{
    public CPerson( string name )
    {
        m_name = name;
    }

    // IPerson
    public string GetName()
    {
        return m_name;
    }

    private string m_name;
}

class CPerson2 : IPerson2
{
    public CPerson2( string name, int age )
    {
        m_name = name;
        m_age = age;
    }

    // IPerson2
    public string GetName() { return m_name; }
```

```
        public int GetAge() { return m_age; }
        private string m_name; private int m_age;
}

public class CApp
{
        public static void Main()
        {
                CPerson bob = new CPerson( "Bob" );
                CPerson2 sheila = new CPerson2( "Sheila", 24 );

                DisplayAge( bob );
                DisplayAge( sheila );
        }

        static void DisplayAge( IPerson person )
        {
                IPerson2 person2 = person as IPerson2;   // QueryInterface lives on !!!
                if( person2 != null )
                        Console.WriteLine( "{0} is {1} years old.", person2.GetName(),
person2.GetAge() );
                else
                        Console.WriteLine( "Sorry, don’t know {0}’s age.", person.GetName() );
        }
}
```

Running the program produces the following output:

```
Sorry, don’t know Bob’s age.
Sheila is 24 years old.
```

44) How can we use the Win32 API from a .NET Framework program? Give sample code

Using platform invoke, .NET Framework programs can access native code libraries by means of static DLL entry points.

Here is an example of C# calling the Win32 MessageBox function:

```
using System;
using System.Runtime.InteropServices;

class MainApp
{
   [DllImport("user32.dll", EntryPoint="MessageBox")]
   public static extern int MessageBox(int hWnd, String strMessage, String strCaption, uint uiType);

   public static void Main()
   {
        MessageBox( 0, "Hello, this is PInvoke in operation!", ".NET", 0 );
    }
}
```

45) Is it true that all C# types derive from a common base class, explain.
Yes and no. All types can be treated as if they derive from object (System.Object), but in order to treat an instance of a value type (e.g. int, float) as object-derived, the instance must be converted to a reference type using a process called 'boxing'. In theory a developer can forget about this and let the run-time worry about when the conversion is necessary, but in reality this implicit conversion can have side-effects that may trip up the unwary.

46) What are the fundamental differences between value types and reference types?
C# divides types into two categories - value types and reference types. Most of the basic intrinsic types (e.g. int, char) are value types. Structs are also value types. Reference types include classes, interfaces, arrays and strings. The basic idea is straightforward - an instance of a value type represents the actual data (stored on the stack), whereas an instance of a reference type represents a pointer or reference to the data (stored on the heap).

The most confusing aspect of this for C++ developers is that C# has predetermined which types will be represented as values, and which will be represented as references. A C++ developer expects to take responsibility for this decision.

For example, in C++ we can do this:

```
int x1 = 3;                // x1 is a value on the stack
int *x2 = new int(3)    // x2 is a reference to a value on the heap
```

but in C# there is no control:

```
int x1 = 3;                // x1 is a value on the stack
int x2 = new int();
x2 = 3;            // x2 is also a value on the stack!
```

47) Why are events not inherited?
When you have an event, you'd usually write a protected member function that fires the event.

```
public class Button
{
public event EventHandler Click;
protected void OnClick(EventArgs e) {
if (Click != null) Click(this, e);
}
}
```

which allows the derived class to fire the event.

The event is private so that other classes can’t walk up to it and mess with the delegate list. If it were protected, any class would be able to mess with the list merely by deriving from the class.

48) What is the Microsoft Intermediate Language (MSIL)? How MSIL is converted to native code

MSIL is the CPU-independent instruction set into which .NET Framework programs are compiled. It contains instructions for loading, storing, initializing, and calling methods on objects.
Combined with metadata and the common type system, MSIL allows for true cross-language integration.
Prior to execution, MSIL is converted to machine code. It is not interpreted.
The JIT compilers in CLR converts the MSIL to the native code

49) Give the uses of System namespace

1) Develop console applications and windows services
2) Define the base data types such as Object, Integer, etc
3) Determine information about the operating system
4) Manage garbage collection
5) Convert and format types
6) Generate random numbers
7) Perform math calculations
8) Define standard system exception
9) Raise and handle events
10) Retrive time zone information

50) Write various System types with their aliases in VB.NET and C#

System type	VB.NET alias	C# alias
System.Byte	Byte	byte
System.Int16	Short	short
System.Int32	Integer	int
System.Int64	Long	long
System.UInt16		ushort
System.UInt32		uint
System.UInt64		ulong