Q.2 a. List and briefly explain the core features of .NET (8)

Answer:

Full interoperability with existing code
Complete and total language integration
A common runtime engine shared by all .NET-aware languages
A base class library
No more COM plumbing
A truly simplified deployment model

Any four features with explaination } 8M
2M each

b. Explain the role of .NET type Metadata. (8)

Answer:

metadata describes each and every type (class, structure, enumeration, and so forth) defined in the binary, as well as the members of each type (properties, methods, events, and so on). It is always the job of the compiler (not the programmer) to emit the latest and greatest type metadata. Because .NET metadata is so wickedly meticulous, assemblies are completely self-describing entities so much so, in fact, that .NET binaries have no need to be registered into the system registry

Explain .Net type Metadata - 4M
Importance of Metabdda - 4M } 8M

Q.3 a. How do you reference external assemblies in .NET? Explain with an example. (8)

Answer:

To illustrate the process of referencing external assemblies, let's update the TestApp application to display aWindows Forms message box. Open your TestApp.cs file and modify it as follows:

```
using System;
// Add this!
using System.Windows.Forms;
class TestApp
{
public static void Main()
{
Console.WriteLine("Testing! 1, 2, 3");
// Add this!
MessageBox.Show("Hello...");
}
    }
```

External assemblies - 4M
Example for illustration - 4M } 8M

b.Explain working with cse.exe response files. (8)

Answer:

C# response files contain all the instructions to be used during the compilation of your current build. By convention, these files end in a *.rsp (response) extension. Assume that you have created a response file named TestApp.rsp that contains the following arguments (as you can see, comments are denoted with the # character):

```
# This is the response file
# for the TestApp.exe app
# of Chapter 2.
# External assembly references.
/r:System.Windows.Forms.dll
# output and files to compile (using wildcard syntax).
/target:exe /out:TestApp.exe *.cs
```

Now, assuming this file is saved in the same directory as the C# source code files to be compiled, you are able to build your entire application as follows (note the use of the @ symbol):

```
csc @TestApp.rsp
```

Explaination with syntax - C# 8M

Q.4 a. Explain processing command line arguments in C# using a suitable C# code. (8)

Answer:

Assume that you now wish to update HelloClass to process possible command-line parameters:

```
// This time, check if you have been sent any command-line arguments.
using System;
class HelloClass
{
public static int Main(string[] args)
{
Console.WriteLine("***** Command line args *****");
for(int i = 0; i < args.Length; i++)
Console.WriteLine("Arg: {0} ", args[i]);
...
}
}
```

Here, you are checking to see if the array of strings contains some number of items using the Length property of System.Array As you loop over each item in the array, its value is printed to the console window.

Command line argument explan - 4M } 8M
Example in C# - 4M

b. Explain static methods and static constructors in C#. (8)

Answer:

Static methods, static data and static constructors expl with examples.

Static Methods - 4M
Static Constructor - 4M
} 8M

Q.5 a. Explain two pillars of OOP: Encapsulation and Inheritance w.r.t. C#. (8)

Answer:

Encapsulation – how well object's internal implementation is hidden

Inheritance – how does the language promote code reuse.

Polymorphism – how do related objects communicate.

Encapsulation - 4M
Inheritance - 4M
} 8M

b. Explain how overloading is implemented in C#. Explain using an example in C#. (8)

Answer:

b. Like other object-oriented languages, C# allows a type to *overload* various methods. Simply put, when a class has a set of identically named members that differ by the number (or type) of parameters, the member in question is said to be *overloaded*. In the Employee class, you have overloaded the class constructor, given that you have provided two definitions that differ only by the parameter set:

```
public class Employee
{
...
// Overloaded constructors.
public Employee(){ }
public Employee(string fullName, int empID, float currPay) {...}
...
}
```

Constructors, however, are not the only members that may be overloaded for a type

Overloading meaning - 2M
Example code in C# - 6M
} 8M

Q.6 a. How can one debug system exceptions using vs .NET? Explain. (8)

Answer:

Page No. 253, Ref. Book Chapter 5

Explanation with diagram 8M

b.What is do you understand by Garbage collection in C#? Explain the significance of System.GC Type. (8)

Answer:

Garbage collection – if the managed heap does not the sufficient memory to allocate a new object, a garbage collection will occur.
System.GC is a base class library type to interact with garbage collector.

Garbage collection in c# - 4M
Significance of System.GC type - 4M } 8M

Q.7 a. What does Icomparable interface specify? Give a C# code for Icomparable interface. (8)

Answer:
Page No. 302, Ref. Book

Icomparable interface - 2M
Implementation of same - 6M } 8M

b. Can we use interfaces in C# as polymorphic agent? Explain using an example. (8)

Answer:
Page No. 284, Ref. Book

Explaination 8M

Q.8 a. How do you define delegates in C#? Give an example program in C# for its implementation. (8)

Answer:
Page No. 326, 330, Ref. Book

Delegates definition & explanation - 3M
Implementation - 5M } 8M

b. How can one invoke methods asynchronously? Explain using C# code. (8)

Answer:
Page No. 343, Ref. Book

Invoking methods asynchronously explan - 3M
C# code - 5M } 8M

Q.9 a. What is GAC? How can one register an assembly using GAC? Explain. (8)

Answer:
Page No. 440, Ref. Book

GAC internals – 4M
Registering using GAC – 4M } 8M

b. Explain the following using C# code wherever required:
(i) Private assembly (ii) Shared assembly (8)

Answer:
Chapter 9, Page No. 425, 430, Ref. Book

(i) – 4M
(ii) – 4M } 8M

TEXT BOOK

I. C# and the .NET Platform, Andrew Troelsen, Second Edition 2003, Dreamtech Press