**Exam 70-483 Programming in C#**

**Manage program flow (25–30%)**

Implement multithreading and asynchronous processing

Use the Task Parallel library (ParallelFor, Plinq, Tasks); create continuation tasks; spawn threads by using ThreadPool; unblock the UI; use async and await keywords; manage data by using concurrent collections

Manage multithreading

Synchronize resources; implement locking; cancel a long-running task; implement thread-safe methods to handle race conditions

Implement program flow

Iterate across collection and array items; program decisions by using switch statements, if/then, and operators; evaluate expressions

Create and implement events and callbacks

Create event handlers; subscribe to and unsubscribe from events; use built-in delegate types to create events; create delegates; lambda expressions; anonymous methods

Implement exception handling

Handle exception types (SQL exceptions, network exceptions, communication exceptions, network timeout exceptions); catch typed vs. base exceptions; implement try-catch-finally blocks; throw exceptions; determine when to rethrow vs. throw; create custom exceptions

Preparation resources

Asynchronous programming with Async and Await (C# and Visual Basic) <http://msdn.microsoft.com/library/vstudio/hh191443.aspx>

Threading (C# and Visual Basic) <http://msdn.microsoft.com/library/ms173178.aspx>

Selection statements (C# reference) <http://msdn.microsoft.com/library/vstudio/676s4xab.aspx>

**Create and use types (25–30%)**

Create types

Create value types (structs, enum), reference types, generic types, constructors, static variables, methods, classes, extension methods, optional and named parameters, and indexed properties; create overloaded and overriden methods

Consume types

Box or unbox to convert between value types; cast types; convert types; handle dynamic types; ensure interoperability with unmanaged code, for example, dynamic keyword

Enforce encapsulation

Enforce encapsulation by using properties, by using accessors (public, private, protected), and by using explicit interface implementation

Create and implement a class hierarchy

Design and implement an interface; inherit from a base class; create and implement classes based on the IComparable, IEnumerable, IDisposable, and IUnknown interfaces

Find, execute, and create types at runtime by using reflection

Create and apply attributes; read attributes; generate code at runtime by using CodeDom and lambda expressions; use types from the System.Reflection namespace (Assembly, PropertyInfo, MethodInfo, Type)

Manage the object life cycle

Manage unmanaged resources; implement IDisposable, including interaction with finalization; manage IDisposable by using the Using statement; manage finalization and garbage collection

Manipulate strings

Manipulate strings by using the StringBuilder, StringWriter, and StringReader classes; search strings; enumerate string methods; format strings

Preparation resources

Types (C# programming guide) <http://msdn.microsoft.com/library/ms173104.aspx>

Classes and structs (C# programming guide) <http://msdn.microsoft.com/library/vstudio/ms173109.aspx>

Object-oriented programming (C# and Visual Basic) <http://msdn.microsoft.com/library/dd460654.aspx>

**Debug applications and implement security (25–30%)**

Validate application input

Validate JSON data; data collection types; manage data integrity; evaluate a regular expression to validate the input format; use built-in functions to validate data type and content out of scope: writing regular expressions

Perform symmetric and asymmetric encryption

Choose an appropriate encryption algorithm; manage and create certificates; implement key management; implement the System.Security namespace; hashing data; encrypt streams

Manage assemblies

Version assemblies; sign assemblies using strong names; implement side-by-side hosting; put an assembly in the global assembly cache; create a WinMD assembly

Debug an application

Create and manage compiler directives; choose an appropriate build type; manage programming database files and symbols

Implement diagnostics in an application

Implement logging and tracing; profiling applications; create and monitor performance counters; write to the event log

Preparation resources

Validating data <http://msdn.microsoft.com/library/vstudio/t3b36awf.aspx>

.NET Framework regular expressions <http://msdn.microsoft.com/library/hs600312.aspx>

**Implement data access (25–30%)**

Perform I/O operations

Read and write files and streams; read and write from the network by using classes in the System.Net namespace; implement asynchronous I/O operations

Consume data

Retrieve data from a database; update data in a database; consume JSON and XML data; retrieve data by using web services

Query and manipulate data and objects by using LINQ

Query data by using operators (projection, join, group, take, skip, aggregate); create method-based LINQ queries; query data by using query comprehension syntax; select data by using anonymous types; force execution of a query; read, filter, create, and modify data structures by using LINQ to XML

Serialize and deserialize data

Serialize and deserialize data by using binary serialization, custom serialization, XML Serializer, JSON Serializer, and Data Contract Serializer

Store data in and retrieve data from collections

Store and retrieve data by using dictionaries, arrays, lists, sets, and queues; choose a collection type; initialize a collection; add and remove items from a collection; use typed vs. non-typed collections; implement custom collections; implement collection interfaces

Preparation resources

File system and the registry (C# programming guide) <http://msdn.microsoft.com/library/vstudio/2kzb96fk.aspx>

Connecting to data in Visual Studio <http://msdn.microsoft.com/library/vstudio/ms171886.aspx>

Editing data in your application <http://msdn.microsoft.com/library/vstudio/ms171928.aspx>

<https://www.microsoft.com/en-us/learning/exam-70-483.aspx>