Set 1

Time:

Attempt All Tasks. **[5*4=20]**

1. Write a C# program that demonstrates the use of constructors and properties in a class called Employee. The class should have properties for Name, ID, and Salary. Implement a constructor that initializes these properties.
2. Create a C# program to demonstrate method overriding. Define a base class Shape with a virtual method Draw(), and override this method in a derived class Circle.
3. Develop a simple ASP.NET Core MVC application that accepts a name through a form and displays a greeting message. Implement model binding for the form submission.
4. Using ADO.NET, write a C# program to establish a connection with a SQL Server database and fetch employee records from a table named Employees. Display the retrieved records in the console.

Set 2

Time:

Attempt All Tasks. **[5*4=20]**

1. Write a C# program demonstrating the use of async and await by implementing an asynchronous method that fetches data from a file and prints it.
2. Create an ASP.NET Core MVC Controller with an action method that returns a JSON response containing a list of products.
3. Implement a basic ASP.NET Core Web API with an endpoint that returns a list of students. Use dependency injection to manage a student service.
4. Implement role-based authorization in an ASP.NET Core MVC application. Ensure that only users in the "Admin" role can access a specific action method.

Set 3

Time:

Attempt All Tasks. **[5*4=20]**

1. Create a C# program to demonstrate the use of delegates and events. Define a delegate that triggers an event when a user logs into a system.
2. Implement an ASP.NET Core application that uses Entity Framework Core to insert and retrieve product details from a database.
3. Write a C# program that demonstrates the use of LINQ to filter a list of students based on their grades.
4. Secure an ASP.NET Core Web API by implementing JWT authentication. Ensure only authenticated users can access the API.

Set 4

Time:

Attempt All Tasks. **[5*4=20]**

1. Write a C# program demonstrating the use of an abstract class and an interface to define common behaviors for different types of bank accounts.
2. Develop an ASP.NET Core MVC application with session state management. Store and retrieve user preferences using session variables.
3. Implement client-side validation in an ASP.NET Core application using jQuery. Validate an email and phone number input field before form submission.
4. Deploy an ASP.NET Core application to IIS or Kestrel and demonstrate how to configure the hosting environment.