ASP.NET MVC:

1. Write short notes on:
    - Action Filters
    - Partial Views
    - Layout Pages in Razor
2. Describe the working of LINQ with examples of:
    - Filtering (where)
    - Projection (select)
    - Aggregation (Sum, Count, Max)
3. Write code to create a controller named StudentController with an action method Index that returns a list of students to the view.
4. Configure a route in RouteConfig.cs to map the URL http://localhost:1234/Product/Details/5 to the ProductController and Details action method with an ID parameter.
5. Write code to perform the following using MVC and Entity Framework:
    - Insert a new employee record into the database
    - Display a list of employees in a view
    - Update employee details
    - Delete an employee record
6. Write Razor syntax to create a form that accepts:
    - Student Name (TextBox)
    - Gender (Radio Buttons)
    - Course Selection (DropDownList)
    - Submit Button
7. Create a strongly typed view that displays student details (Id, Name, Age) passed from a controller.
8. Write code for a form that submits student registration details to a controller, and the controller should display the submitted details in a confirmation view.

ASP.NET Core:

1. With a neat diagram, explain the MVC pattern in ASP.NET Core. Describe the project layout of an ASP.NET Core MVC application.
2. Explain the role of Middleware in ASP.NET Core. Provide examples.
3. Explain how Routing works in ASP.NET Core MVC.
4. What are View Components in ASP.NET Core? How are they different from Partial Views?
5. Explain how Entity Framework Core is used for CRUD operations in ASP.NET Core.
6. Write code for a Model class Student with properties Id, Name, Course, and Marks. Show how to pass this model data from Controller to View.
7. Write an Action Method that returns a list of employees from a Controller to a View.
8. Implement CRUD operations in ASP.NET Core MVC for a Product entity (Id, Name, Price, Quantity).
9. Write code to configure Entity Framework Core with a database connection in Startup.cs.

10. Create a form using Razor syntax in a View to collect user registration data and pass it to the Controller.

Entity Framework:

1. How does EF handle relationships between entities (one-to-one, one-to-many, many-to-many)?
2. Write a simple Student and Course model using Code-First with a one-to-many relationship.
3. What are migrations in EF Code-First? How do you create and apply them?
4. How does EF Core support multiple database providers? Give examples.
5. Compare Code-First, Database-First, and Model-First approaches. Which would you choose in a greenfield project and why?
6. Suppose you are designing an E-commerce system. Show how you would design Product, Category, and Order entities using EF Code-First. Implement CRUD operations for Product.
7. Write C# code to show how LINQ-to-Entities can be used to fetch all employees whose salary is greater than 50,000.
8. In EF Core, how would you configure a composite key for a table using Fluent API? Provide code.
9. Explain with code how to enable and disable lazy loading in EF Core.
10. Design a simple Student Management System using EF Code-First with Student, Course, and Enrollment entities. Write queries to fetch:
    - All students with their enrolled courses.
    - Students who are not enrolled in any course.
    - The total number of enrollments in each course.