# Advanced ASP.NET Core Program Questions

## Lab 1: Advanced Variables, Data Types, Operators

1. 1. Write a program to calculate the compound interest using user-defined functions, with parameters for principal, rate, time, and frequency of compounding.
2. 2. Develop a program to accept a string and check if it is a valid email address using regular expressions.
3. 3. Create a program to accept three numbers and determine if they can form the sides of a triangle. If yes, classify the triangle as equilateral, isosceles, or scalene.
4. 4. Write a program to convert a binary number to its decimal equivalent without using in-built functions.
5. 5. Develop a program to perform bitwise operations (AND, OR, XOR) and display their results along with the binary representation of the input numbers.

## Lab 2: Advanced Class and Object, Constructors, Inheritance

1. 1. Create a class "Product" with attributes ProductID, Name, Price, and Discount. Include a method to calculate the net price based on discount. Use a parameterized constructor for initialization.
2. 2. Design a "Library" system where a class "Book" has attributes like BookID, Title, Author, and Issued (true/false). Create a derived class "IssuedBook" that keeps track of the issue date and return date.
3. 3. Create a class "Shape" with a virtual method Area(). Derive classes "Circle", "Square", and "Rectangle" that override the Area() method to calculate specific areas.
4. 4. Implement a class "Customer" with attributes Name, Email, and an array of Orders. Use a constructor to initialize and display customer details along with their orders.
5. 5. Develop a program using multi-level inheritance: a base class "Person", a derived class "Student", and a further derived class "Graduate". Include constructors and demonstrate method overriding.

## Lab 3: Advanced Exception Handling, Interface, Abstraction, String Functions

1. 1. Write a program to handle a custom exception "InvalidAgeException". Throw this exception if the user enters an age outside the range of 0 to 120.
2. 2. Create an abstract class "Account" with methods for deposit and withdrawal. Derive "SavingsAccount" and "CurrentAccount" classes that implement these methods with specific rules.
3. 3. Write a program to implement an interface "IShape" with methods for calculating perimeter and area. Implement this interface in classes "Circle" and "Triangle".
4. 4. Develop a program to accept a paragraph and find the frequency of each word using a dictionary.
5. 5. Write a program to check if a given string is a valid palindrome, ignoring spaces, punctuation, and case.

## Lab 4: Advanced Method Overloading, Method Overriding

1. 1. Implement a class "Bank" with methods Deposit() and Withdraw() overloaded to accept cash, check, or online transfer details. Demonstrate method overloading.
2. 2. Create a class "Employee" with a virtual method DisplayDetails(). Override this method in derived classes "Manager" and "Technician" to display specific details.
3. 3. Design a class "Vehicle" with methods Start() and Stop() overloaded to accept optional parameters like key type or remote ID. Demonstrate method overloading and default parameters.
4. 4. Write a program using method overriding to simulate a school system with a base class "Person" and derived classes "Teacher" and "Student". Include virtual and overridden methods for DisplayDetails().
5. 5. Develop a class "Company" with overloaded constructors for different initialization scenarios: with only name, with name and location, and with name, location, and number of employees.

## Lab 5: Advanced Collection Classes

1. 1. Create a program to manage a "Product Inventory" using a List collection. Include methods for adding, updating, and removing products.
2. 2. Develop a "Task Scheduler" using a Dictionary where the key is the task date and the value is the task description. Include functionality to display tasks for a specific date.
3. 3. Write a program to use a Stack to reverse a sentence entered by the user.
4. 4. Create a "Library Management" system using a Queue where books are issued in FIFO order. Include methods for adding and issuing books.
5. 5. Develop a program using a Hashtable to store employee details with EmployeeID as the key and EmployeeName as the value. Include methods to search by ID and display all employees.