Lab	Type	Practical
1		**Variables, Data Types, Operators**
	A	1. Write a program to print your name, address, contact number & city.
	A	2. Write a program to get two numbers from user and print those two numbers.
	A	3. Write program to prompt a user to input his/her name and country name and then output will be shown as given: Hello <yourname> from country <countryname>
	B	4. Write a C# program that allows the user to convert temperature from Celsius to Fahrenheit and vice versa. The user should be able to select the conversion type and input the temperature value to receive the converted result.
	B	5. Write a program to compute an employee's gross and net salary. The program should take basic salary as input and calculate HRA (10%), DA (15%), and deductions (8%) to display gross and net salary.
	B	6. Create a C# program that calculates area and perimeter of geometric shapes such as rectangle, circle, and triangle. The program should allow the user to choose the shape and input the required dimensions.
	B	7. Write a C# program that accepts marks in five subjects, calculates total and percentage, and displays a grade (A/B/C/Fail) based on percentage criteria. (Assign grade: A ≥ 75, B ≥ 60, C ≥ 45, else Fail.)
	C	8. Develop a program to convert a given amount in Indian Rupees to other currencies (USD, EUR, GBP). Assume fixed exchange rates and display the converted value with labels.
	C	9. During a festive season, an online shopping platform offers tiered discounts: a. 5% for orders below ₹5000 b. 10% for orders between ₹5000 and ₹10000 c. 15% for orders above ₹10000 Write a C# program that accepts the total purchase amount and calculates the discount based on the above conditions. The program should also display the original amount, discount amount, and final payable amount.

C

10. A travel company provides cab services with the following conditions:

Vehicle Type	Rate per Km	Driver Allowance (if distance > 150 km)
Sedan	₹12/km	₹500
SUV	₹15/km	₹700
Luxury	₹20/km	₹1000

In addition, a fuel surcharge of 5% is added to every trip. If the distance exceeds 100 km, a 10% discount is applied to the fare before adding the surcharge and driver allowance.
The company also allows users to choose one-way or round-trip journeys (round-trip doubles the distance).

Write a C# program that calculates and displays:

- Total distance (considering trip type)
- Base fare
- Discount (if applicable)
- Fuel surcharge
- Driver allowance (if applicable)
- Final payable fare

2		**<u>Conditions & Looping</u>**
	A	1. Write a C# program to print the multiplication table of a given number using a for loop.
	A	2. Develop a program that counts how many digits, alphabets, and special characters are present in an input string.
	A	3. Write a C# program to calculate the grade of a student based on total marks entered. Use if-else if conditions to classify grades as A, B, C, D, or Fail.
	B	4. Create a C# program that takes an integer limit and calculates the sum of all even and odd numbers separately using a loop.
	B	5. Write a C# program that computes the factorial of a number using a for loop.
	C	6. Design a C# program that checks whether a password entered by the user is strong. The password is considered strong if it has at least 8 characters, one uppercase letter, one lowercase letter, one digit, and one special character.
	C	7. Write a C# program to check whether a given number is a prime number or not using a for loop and conditional statements.
	C	8. Develop a program that takes an integer input and reverses its digits using a while loop.
	C	9. Create a program to check whether a given number is a palindrome.
	C	10. Write a program to display the first *n* terms of the Fibonacci series using a for loop.

3		**Class, Object, Constructors & Exception Handling**
	A	1. Write a C# program that defines a Student class with data members Name, RollNo, and Marks. Create objects for two students and display their details using a class method.
	A	2. Define a class Rectangle with data members length and breadth. Use a parameterized constructor to initialize them and a method to calculate and return the area.
	A	3. Write a program that accepts two numbers from the user and divides the first number by the second. Use try-catch to handle the divide-by-zero exception gracefully.
	B	4. Design a BankAccount class with members accountNumber, holderName, and balance. Include methods for deposit and withdrawal. Ensure that withdrawal should not allow balance to go below zero, and use exception handling for invalid transactions.
	B	5. Write a C# program to demonstrate constructor overloading by creating a Person class with multiple constructors that accept different numbers of parameters. Display the initialized values.
	B	6. Create a class Employee with members empID, empName, and salary. Initialize these values using a parameterized constructor and display them using a class method.
	C	7. Develop a ShoppingCart class for an online store. Each item has a name, price, and quantity. The cart should calculate the total bill and throw an exception if any item's quantity is zero or negative.
	C	8. Create a CarRental class that stores carModel, dailyRate, and rentedDays. The constructor should initialize all values. Include a method CalculateRent() that multiplies rate by days. Handle an exception if days are negative or zero.
	C	9. Design a FlightTicket class for an airline booking system. The constructor should take passengerName, flightNumber, and ticketPrice. If the ticket price is below ₹500, throw an exception indicating "Invalid Ticket Price." Display booking details if valid.

4		**<u>Method Overloading, Overriding & Access Modifiers</u>**
	A	1. Write a C# program that defines a class Calculator with overloaded methods Add() for adding two integers, three integers, and two double values. Demonstrate method overloading by calling each version.
	A	2. Create a class Employee that uses method overloading to display employee information. One method should take only name, another should take name and age, and another should take all three: name, age, and salary.
	A	3. Write a C# program that defines a class Person with members that use different access modifiers (public, private, protected, internal). Show how these members can and cannot be accessed from different parts of the program — inside the class, from a derived class, and from an external class.
	B	4. Create a base class Animal with a virtual method Sound(). Derive classes Dog and Cat and override the Sound() method in each to print unique messages. Demonstrate runtime polymorphism.
	B	5. Design a base class Shape with a virtual method CalculateArea(). Derive Circle, Rectangle, and Triangle classes that override this method to compute area based on their own formulas.
	C	6. Create a class BankTransaction with overloaded Transfer() methods to transfer money between two accounts (using amount only) and between two accounts with an additional message (using amount and description). Demonstrate method overloading and use private fields with public access methods.
	C	7. Develop a LibraryItem base class with Title and Author. Derive two subclasses — Book and Magazine. Override a method DisplayInfo() to show details specific to each type. Demonstrate method overriding and use access modifiers to restrict direct field modification.
	C	8. Design a billing system using polymorphism. Create a base class Customer with method CalculateBill() and derived classes RegularCustomer and PremiumCustomer that override it with different discount logics. Demonstrate access modifiers for secure data access.

5		**Inheritance, Interface & Abstraction**
	A	1. Write a C# program to demonstrate single inheritance. Create a base class Animal with a method Eat(), and a derived class Dog that adds a method Bark(). Show how both methods can be accessed through the derived class.
	A	2. Demonstrate multilevel inheritance by creating three classes: Vehicle, Car, and ElectricCar. Each class should have a method displaying its type. Use an object of ElectricCar to show access to all levels.
	A	3. Create a base class Shape with a virtual method Area(). Derive classes Circle and Rectangle that override this method to calculate their respective areas. Display results using polymorphism.
	B	4. Create an abstract class Appliance with abstract method TurnOn(). Derive Fan and Light classes that provide specific implementations of this method. Demonstrate abstraction by calling methods using base class reference.
	B	5. Create an interface IPrintable with a method PrintDetails(). Implement this interface in two classes: Book and Magazine. Each should display different information.
	B	6. Define two interfaces IMovable and ISound. Create a class Robot that implements both interfaces, providing methods for Move() and MakeSound(). Demonstrate multiple interface implementation in C#.
	C	7. Develop an online payment system using abstraction and inheritance. Create an abstract class Payment with abstract method MakePayment(). Derive classes CreditCardPayment and UPIPayment that implement this method differently. Use exception handling if the amount entered is less than ₹100.
	C	8. Create an abstract class Employee with properties Name, Salary, and abstract method CalculateBonus(). Implement two subclasses Manager and Developer where each calculates a bonus differently (Manager → 20%, Developer → 10%). Demonstrate polymorphism using base class references.
	C	9. Design a vehicle rental system using interfaces. Create an interface IRentable with methods CalculateRent() and DisplayDetails(). Implement this interface in Car and Bike classes, each with different rent-per-day logic. Use a list to manage multiple rentals.

6		**Collection Classes & Strings**
	A	1. Create a program that simulates a simple "Recent Tasks" tracker using a Stack<string>. Each time the user performs a task, push it onto the stack. The program should display the most recent task (top of the stack) and allow "undo" by popping an item.
	A	2. Simulate a customer service queue using a Queue<string>. Customers arrive in sequence and are served in the order they arrived (FIFO). The program should allow adding customers, serving one, and displaying who's next.
	A	3. Write a C# program to count the number of vowels and consonants in a given string. Ignore spaces and handle both uppercase and lowercase characters.
	A	4. Write a C# program to check whether a given string is a palindrome (reads the same backward and forward). Ignore case and spaces.
	B	5. Write a C# program to count how many times each word appears in a given sentence. Use string methods and looping structures to process the input.
	B	6. Create a List<string> to manage a shopping list. The user can add items, remove items, and view the complete list. If an item already exists, do not add it again.
	B	7. Write a C# program using a Dictionary<string, int> to count the number of occurrences of each word in a sentence entered by the user.
	B	8. A company wants to store unique email addresses of subscribers. Write a program using HashSet<string> that accepts multiple email entries and ensures no duplicates are added. Display the total number of unique emails stored.
	C	9. A library wants to maintain records of borrowed books using a Dictionary<string, Queue<string>>, where the key is the book title, and the value is a queue of borrower names (in order of borrowing). Write a program that allows adding borrowers and viewing who borrowed which books.
	C	10. A hospital uses two queues — one for **Normal patients** and one for **Emergency patients**. Emergency patients should always be served first. Use two Queue<string> objects to simulate this system.
7		**Design a Static Web using Bootstrap**
	A	Design web pages using given images with the help of bootstrap.
8		**Theme Conversion**
	A	Multiple page admin theme conversion with required pages.

9		**Razor Syntax Overview and Data Passing Techniques**
	A	Print table of 5 using Razor.
	A	Prepare a page which displays student details and his/her semester wise SPI in table format.
	B	Prepare semester wise SPI table data in controller file and store it in ViewBag/ViewData. Display the data to view page using foreach loop.
	B	Use TempData to pass data between two controller actions or from a controller to a view, demonstrating how data persists for a single request or redirect. Display the transferred message or value in the destination view.
10		**Working with Areas & IAction Result object**
	A	Implementing Feature-Based Modularization using Areas for Logical Separation of Application Components (Admin, Manager, Employee)
	B	Demonstrate different types of action results by implementing controller methods that return a View, Content, JSON, File, Redirect, and Status Code using IActionResult. Test each action through browser links or buttons to observe the different response types.
11		**Create Database and prepare stored procedures for Select command**
	A	Create Database : AddressBook also create SelectAll and SelectByPK stored procedure for Country, State and City tables.
	B	Pass the City Name in procedure and display all the records based on city name.
	B	Pass the State Name in procedure and display all the cities belongs to the that state.
	C	Pass the Country name in procedure and display all the states with number of cities.
12		**Prepare stored procedure for Insert, Update and Delete command**
	A	Create all tables Insert, Update and Delete stored procedures for Country, State and City tables.
	B	Create an Insert, Update & Delete Stored Procedure for your own one table with minimum 6-7 columns.
13		**Prepare Layout Page**
	A	Design Layout page that describes the overall UI, Add views for Header and Footers as required.
14		**Prepare Design Pages and Apply Routing**
	A	Design List Page & Add/Edit Pages. Also Apply Attribute Routing between all the pages. Use appropriate routing attributes as required.

15	**A** **A** **B** **C**	**Implementation of Html Helpers** Student registration form using Standard html helpers. (StudentName, Branch, Semester, Birthdate, Mobile, Email, Address, City, Hobbies, Gender) Student registration form using Strongly typed html helpers. (StudentName, Branch, Semester, Birthdate, Mobile, Email, Address, City, Hobbies, Gender) Employee Registration form using Standard html helpers. Job Inquiry form using Strongly typed html helpers.
16	**A** **B**	**Building Custom Tag Helpers** Create all alerts as Custom tag helper for Success, Warning & Info. Build a tag helper that will render HTML that generates a link that allows the user to send an email to the owner of project.
17	**A**	**Partial Views** Use Partial Views to divide a web page into reusable sections such as a header, footer, or navigation bar. Render these partial views within the main view.
18	**A**	**Model creation and Data annotation** Prepare model classes as per requirement and Implement data annotation on all the model classes.
19	**A**	**Apply Server side validation** Apply server side validation for all the submit requests.
20	**A** **B**	**Working with Form Collection and Bind Attribute** Developing a Feedback Form using Form Collection to Access and Display User Inputs.(Form fields - Id, Name, Email, Subject, Observation) Implementing Selective Model Binding using the [Bind] Attribute to Secure Model Properties for Student Model (Id, Name, Email, Password)
21	**A** **B**	**Demonstration of File Upload** Design a view by which user can upload his/her resume to the server and display the uploaded resume. Design a view from user can upload their profile picture.
22	**A**	**Dynamic dashboard design – I** Design dynamic dashboard to display different summaries and statistics in tabular format.

23	**A**	**Dynamic dashboard design – II** Design dynamic dashboard to visualize data in different types of charts.
24	**A**	**Database connectivity and Implementation of Read operation** Create database connectivity and Display data (All Records).
25	**A**	**Implementation of Insert and Delete functionality** Apply Insert and Delete record functionality
26	**A**	**Implementation of Update functionality** Apply Update record functionality
27	**A**	**Implement Dropdown functionality** Implement dropdown functionality as required.
28	**A**	**Implementation of Search functionality** Implement Search functionality for all the list pages.
29	**A** **B**	**Login and User Registration operation** Implement Login functionality. Implement User Registration functionality.
30	**A** **B**	**Implementation of Excel Export functionality** Implement Search functionality for all the list pages. Add a button by which user can export table data to excel.