Q1. Create a class BankAccount with a property Balance.

• The property should allow deposit but not allow direct withdrawal (only decrease balance via a method)

. • Demonstrate depositing 1000 and withdrawing 500 using the property and method

```
using System;


class BankAccount
{
   private decimal balance;


   public decimal Balance
   {
      get { return balance; }
      private set { balance = value; }  // deposit allowed internally
   }


   public void Deposit(decimal amount)
   {
      Balance += amount;
   }


   public void Withdraw(decimal amount)
   {
      if (amount <= Balance)
         Balance -= amount;
```

```
  else
        Console.WriteLine("Insufficient balance!");
    }
}

class Program1
{
    static void Main()
    {
        BankAccount acc = new BankAccount();
        acc.Deposit(1000);
        Console.WriteLine("After deposit: " + acc.Balance);

        acc.Withdraw(500);
        Console.WriteLine("After withdrawal: " + acc.Balance);
    }
}
```

. Q2. Design a class Student with a property Age.

• Ensure that only values between 5 and 25 are allowed.

• If invalid age is set, default to 18. • Show the behavior for age 4, 20, and 30.

```
using System;

class Student
{
    private int age;
```

```
    public int Age
    {
        get { return age; }
        set
        {
            if (value >= 5 && value <= 25)
                age = value;
            else
                age = 18; // default
        }
    }
}

class Program2
{
    static void Main()
    {
        Student s1 = new Student { Age = 4 };
        Console.WriteLine("Age set to 4 → " + s1.Age);
        Student s2 = new Student { Age = 20 };
        Console.WriteLine("Age set to 20 → " + s2.Age);
 Student s3 = new Student { Age = 30 };
        Console.WriteLine("Age set to 30 → " + s3.Age);
    }
}
```

Q3. Create a class Employee with:

• A private field basicSalary.

• A read-only property TotalSalary that calculates salary with 20% bonus

. • Demonstrate setting basicSalary = 30000 and display TotalSalary.

```
using System;
class Employee
{
   private double basicSalary;
   public double BasicSalary
   {
      get { return basicSalary; }
      set { basicSalary = value; }
   }
   public double TotalSalary
   {
      get { return basicSalary + (0.2 * basicSalary); } // 20% bonus
   }
}
class Program3
{
   static void Main()
   {
      Employee emp = new Employee();
      emp.BasicSalary = 30000;
```

```
        Console.WriteLine("Total Salary: " + emp.TotalSalary);
    }
}
```

Q4. Build a class Product with two auto-properties: Price and Discount.

• Add a method to calculate the final price = Price – (Price * Discount/100).

• Show result for a product with Price = 2000 and Discount = 10%.

```
using System;
class Product
{
    public double Price { get; set; }
    public double Discount { get; set; }
    public double FinalPrice()
    {
        return Price - (Price * Discount / 100);
    }
}
class Program4
{
    static void Main()
    {
        Product p = new Product { Price = 2000, Discount = 10 };
        Console.WriteLine("Final Price: " + p.FinalPrice());
    }
}
```

Q5. Create a Car class with a property Speed.

- **Speed should not exceed 180 km/h; if it exceeds, reset to 180.**
- **Write code to set speed = 150, then 200, and display the final speed.**

```
using System;

class Car
{
    private int speed;

    public int Speed
    {
        get { return speed; }
        set
        {
            if (value > 180)
                speed = 180;
            else
                speed = value;
        }
    }
}

class Program5
{
```

```
    static void Main()
    {
        Car c = new Car();
        c.Speed = 150;
        Console.WriteLine("Speed set to 150 → " + c.Speed);

        c.Speed = 200;
        Console.WriteLine("Speed set to 200 → " + c.Speed);
    }
}
```

Q6. Define a delegate Operation for performing arithmetic operations.

- **Implement methods Add and Subtract.**
- **Ask the user for two numbers and apply both operations using the delegate.**

```
using System;

delegate int Operation(int a, int b);

class Program6
{
    static int Add(int x, int y) => x + y;
    static int Subtract(int x, int y) => x - y;

    static void Main()
    {
```

```
        Console.Write("Enter first number: ");
        int n1 = int.Parse(Console.ReadLine());

        Console.Write("Enter second number: ");
        int n2 = int.Parse(Console.ReadLine());

        Operation op = Add;
        Console.WriteLine("Addition: " + op(n1, n2));

        op = Subtract;
        Console.WriteLine("Subtraction: " + op(n1, n2));
    }
}
```

Q7. Create a delegate FormatText that accepts a string.

- **Implement methods to return:**

1. The string in uppercase.

2. The string in lowercase.

- **Demonstrate calling both through the delegate on input "Hello World".**

```
using System;

delegate string FormatText(string input);

class Program7
{
```

```
    static string ToUpperCase(string s) => s.ToUpper();
    static string ToLowerCase(string s) => s.ToLower();

    static void Main()
    {
        string input = "Hello World";

        FormatText ft = ToUpperCase;
        Console.WriteLine("Uppercase: " + ft(input));

        ft = ToLowerCase;
        Console.WriteLine("Lowercase: " + ft(input));
    }
}
```

Q8. Create a delegate BillingOperation that accepts a product amount.

• Implement four related methods:

1. **ShowTotal → Display original price**
2. **. ApplyDiscount → Apply 10% discount.**
3. **AddTax → Add 18% GST on discounted price.**
4. **FinalBill → Display final payable amount. • Use delegate chaining to call these methods step by step for an item worth ₹5000.**

```
using System;

delegate double BillingOperation(double amount);

class Program8
{
    static double ShowTotal(double amount)
    {
        Console.WriteLine("Original Price: " + amount);
```

```
        return amount;
    }

    static double ApplyDiscount(double amount)
    {
        double discounted = amount - (amount * 0.10);
        Console.WriteLine("After Discount: " + discounted);
        return discounted;
    }

    static double AddTax(double amount)
    {
        double taxed = amount + (amount * 0.18);
        Console.WriteLine("After Tax: " + taxed);
        return taxed;
    }

    static double FinalBill(double amount)
    {
        Console.WriteLine("Final Bill: " + amount);
        return amount;
    }

    static void Main()
    {
        double price = 5000;

        double amt = ShowTotal(price);
        amt = ApplyDiscount(amt);
        amt = AddTax(amt);
        FinalBill(amt);
    }
}
```

Q9. Define a delegate ConvertTemperature that takes double input.

• Implement two methods:

1. Celsius to Fahrenheit.

2. Celsius to Kelvin. • Show result for 25°C.

```
using System;

delegate double ConvertTemperature(double celsius);

class Program9
{
    static double ToFahrenheit(double c) => (c * 9 / 5) + 32;
    static double ToKelvin(double c) => c + 273.15;

    static void Main()
    {
        double celsius = 25;

        ConvertTemperature ct = ToFahrenheit;
        Console.WriteLine("25°C in Fahrenheit: " + ct(celsius));

        ct = ToKelvin;
        Console.WriteLine("25°C in Kelvin: " + ct(celsius));
    }
}
```

Q10. Design a delegate Notifier for sending notifications.

- **Implement two methods: SendEmail and SendSMS.**
- **Call both methods using delegate chaining for the message "Assignment Submitted Successfully".**

```
using System;

delegate void Notifier(string message);

class Program10
{
   static void SendEmail(string msg) => Console.WriteLine("Email Sent: " + msg);
   static void SendSMS(string msg) => Console.WriteLine("SMS Sent: " + msg);

   static void Main()
   {
      Notifier notify = SendEmail;
      notify += SendSMS;

      notify("Assignment Submitted Successfully");
   }
}
```