

Lab Assignment 2 in C#

Q 1. A software company is developing a user profile management system. Design a class UserProfile with private fields like username, password, and email.

- Provide public methods to set and get the values securely.
- Add validation in setters (e.g., email must contain @, password must be at least 6 characters).
- Test your class with at least two objects.

Ans 1. `using System;`

`namespace LabAssignment2`

`{`

`class Userflrofile`

`{`

`private string username;`

`private string password;`

`private string email;`

`public Userflrofile(string username, string password, string email)`

`{`

`SetUsername(username);`

`Setflassword(password);`

`SetEmail(email);`

`}`

`public void SetUsername(string username)`

`{`

`if (string.IsNullOrEmpty(username))`

`{`

`Console.WriteLine("✗ Username cannot be empty.");`

`}`

`else`

`{`

`this.username = username;`

`}`

`}`

`public void Setflassword(string password)`

`{`

`if (password.Length < 6)`

`{`

`Console.WriteLine("✗ flassword must be at least 6 characters`

`long.");`

`}`

`else`

`{`

`this.password = password;`

`}`

`}`

`public void SetEmail(string email)`

`{`

```

        if (!email.Contains("@") || !email.Contains("."))
        {
            Console.WriteLine("✗ Invalid email format.");
        }
        else
        {
            this.email = email;
        }
    }

    public string GetUsername() => username;
    public string GetEmail() => email;

    public string Getflassword()
    {
        if (password != null)
            return new string('*', password.Length);
        else
            return "Not Set";
    }

    public void DisplayInfo()
    {
        Console.WriteLine($"Username: {GetUsername()}");
        Console.WriteLine($"Email: {GetEmail()}");
        Console.WriteLine($"flassword: {Getflassword()}");
        Console.WriteLine(new string('-', 30));
    }
}

class flrogram
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== USER fIROFILE MANAGEMENT SYSTEM ===\n");

        Console.WriteLine("Enter details for User 1:");
        Console.Write("Username: ");
        string u1 = Console.ReadLine();
        Console.Write("flassword: ");
        string p1 = Console.ReadLine();
        Console.Write("Email: ");
        string e1 = Console.ReadLine();

        Userflrofile user1 = new Userflrofile(u1, p1, e1);

        Console.WriteLine("\nEnter details for User 2:");
        Console.Write("Username: ");
        string u2 = Console.ReadLine();
        Console.Write("flassword: ");
        string p2 = Console.ReadLine();
        Console.Write("Email: ");
        string e2 = Console.ReadLine();

        Userflrofile user2 = new Userflrofile(u2, p2, e2);

        Console.WriteLine("\n=== USER fIROFILES ===");
        Console.WriteLine("User 1:");

```

```

        Console.WriteLine("User 2:");
        user2.DisplayInfo();

        Console.WriteLine("frogram finished successfully!");
        Console.WriteLine("Developed by Manish Parmar.");
    }
}
}

```

The screenshot displays a Windows 11 desktop environment. The primary focus is the Visual Studio Code (VS Code) application, which is open with a C# project named 'labassign'. The editor window shows a file named 'Program.cs' containing a C# program for a 'User Profile Management System'. The code includes prompts for Username, Email, and Password, and provides feedback for invalid input (e.g., 'Username cannot be empty!', 'Password must be at least 6 characters long!', 'Invalid email! It must contain '@'.'). The program also displays 'User Profile Details' and a 'Developed by Manish Parmar' message.

The Output window at the bottom of the VS Code interface shows the program's execution. It displays the prompts and user input, followed by the program's feedback messages. The output concludes with 'Developed by Manish Parmar' and a message indicating the program exited with code 0 (0x0). The bottom status bar of VS Code shows 'Build succeeded'.

The Windows Taskbar at the bottom of the screen shows the system clock as 12:12 PM on 10/24/2025. The taskbar includes icons for the Start menu, Search, and several open applications, including VS Code, a file explorer, and a web browser. The system tray on the right shows network, volume, and battery status icons.

- Create a base class `Vehicle` with properties like `Make`, `Model`, and `Year`.
- Create derived classes `Truck` and `Bus`, each having a method `DisplayDetails()`.

```
{
    class Vehicle
    {
        public string Make { get; set; }
        public string Model { get; set; }
        public int Year { get; set; }

        public Vehicle(string make, string model, int year)
        {
            Make = make;
            Model = model;
            Year = year;
        }
    }
}
```

```

    }

    public virtual void DisplayDetails()
    {
        Console.WriteLine($"Make: {Make}, Model: {Model}, Year: {Year}");
    }
}

class Truck : Vehicle
{
    public double LoadCapacity { get; set; }

    public Truck(string make, string model, int year, double loadCapacity)
        : base(make, model, year)
    {
        LoadCapacity = loadCapacity;
    }

    public override void DisplayDetails()
    {
        Console.WriteLine("=== Truck Details ===");
        base.DisplayDetails();
        Console.WriteLine($"Load Capacity: {LoadCapacity} tons");
        Console.WriteLine(new string('-', 30));
    }
}

class Bus : Vehicle
{
    public int SeatingCapacity { get; set; }

    public Bus(string make, string model, int year, int seatingCapacity)
        : base(make, model, year)
    {
        SeatingCapacity = seatingCapacity;
    }

    public override void DisplayDetails()
    {
        Console.WriteLine("=== Bus Details ===");
        base.DisplayDetails();
        Console.WriteLine($"Seating Capacity: {SeatingCapacity} passengers");
        Console.WriteLine(new string('-', 30));
    }
}

class fIrogram
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== TRAN$FIORT COMFIANY VEHICLE SYSTEM ===\n");

        Truck truck1 = new Truck("Volvo", "FH16", 2022, 18.5);

        Bus bus1 = new Bus("Mercedes-Benz", "Citaro", 2021, 50);
    }
}

```

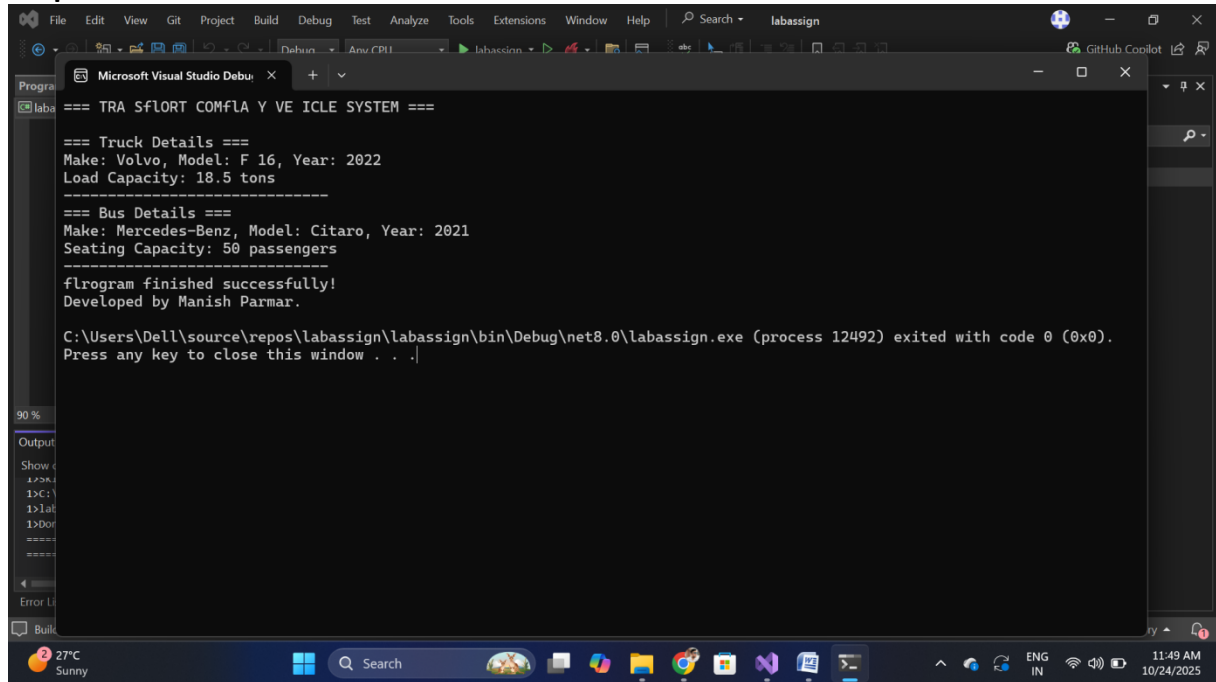
```

truck1.DisplayDetails();
bus1.DisplayDetails();

Console.WriteLine("frogram finished successfully!");
Console.WriteLine("Developed by Manish Parmar.");
}
}
}

```

Output:



```

Microsoft Visual Studio
labassign

Program Output
labassign
=== TRA SFLORT COMFLA Y VE ICLE SYSTEM ===

=== Truck Details ===
Make: Volvo, Model: F 16, Year: 2022
Load Capacity: 18.5 tons
=====
=== Bus Details ===
Make: Mercedes-Benz, Model: Citaro, Year: 2021
Seating Capacity: 50 passengers
=====
frogram finished successfully!
Developed by Manish Parmar.

C:\Users\Dell\source\repos\labassign\labassign\bin\Debug\net8.0\labassign.exe (process 12492) exited with code 0 (0x0).
Press any key to close this window . . .

```

Q 3. Build a calculator class that can perform basic operations using method overloading.

- Implement multiple Add() methods that take different numbers and types of parameters (int, float, double).
- Write a program to test all the variations of the Add() method.

Ans 3. using System;

```

namespace LabAssignment2
{
    class Calculator
    {
        public int Add(int a, int b)
        {
            return a + b;
        }

        public int Add(int a, int b, int c)
        {
            return a + b + c;
        }
    }
}

```

```

    public float Add(float a, float b)
    {
        return a + b;
    }

    public double Add(double a, double b)
    {
        return a + b;
    }

    public double Add(int a, double b)
    {
        return a + b;
    }
}

class flrogram
{
    static void Main(string[] args)
    {
        Calculator calc = new Calculator();

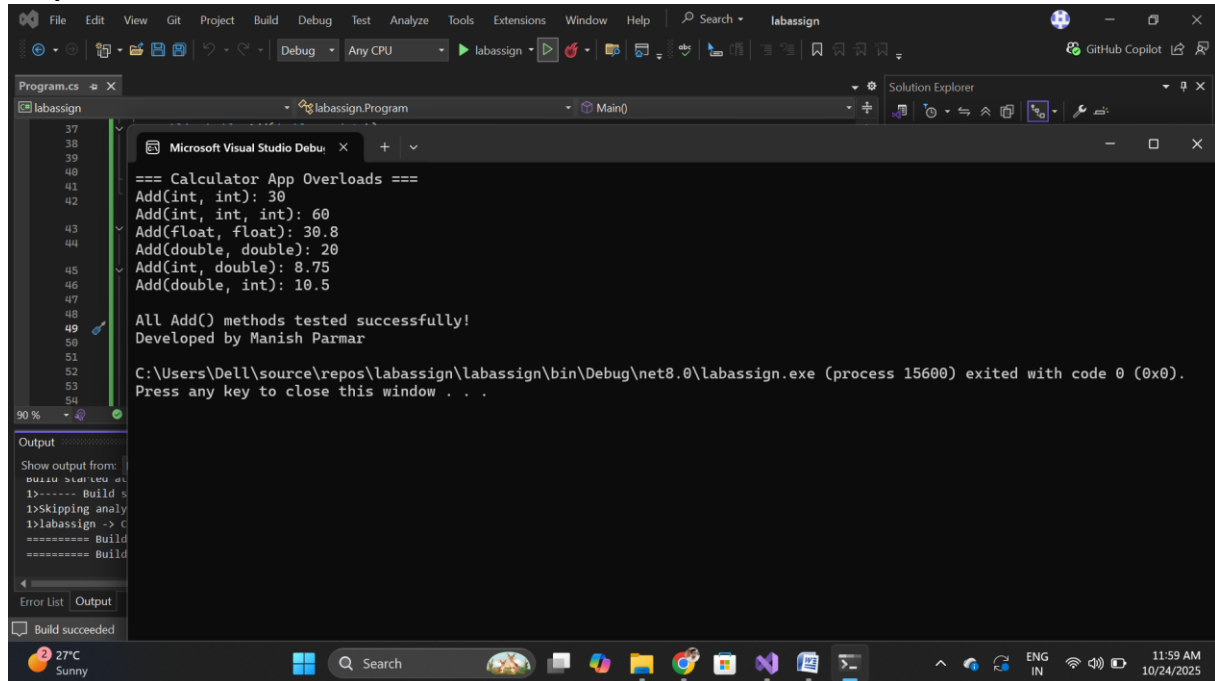
        Console.WriteLine("=== CALCULATOR Aflfl (METHOD OVERLOADING DEMO)
===\n");

        Console.WriteLine($"Add(int, int): 5 + 10 = {calc.Add(5, 10)}");
        Console.WriteLine($"Add(int, int, int): 2 + 4 + 6 = {calc.Add(2, 4,
6)}");
        Console.WriteLine($"Add(float, float): 3.5 + 2.5 = {calc.Add(3.5f,
2.5f)}");
        Console.WriteLine($"Add(double, double): 5.75 + 4.25 =
{calc.Add(5.75, 4.25)}");
        Console.WriteLine($"Add(int, double): 10 + 2.5 = {calc.Add(10,
2.5)}");

        Console.WriteLine("\nflrogram finished successfully!");
        Console.WriteLine("Developed by Manish Parmar");
    }
}

```

Output:



```
Microsoft Visual Studio Debug Console
=== Calculator App Overloads ===
Add(int, int): 30
Add(int, int, int): 60
Add(float, float): 30.8
Add(double, double): 20
Add(int, double): 8.75
Add(double, int): 10.5

All Add() methods tested successfully!
Developed by Manish Parmar

C:\Users\Dell\source\repos\labassign\labassign\bin\Debug\net8.0\labassign.exe (process 15600) exited with code 0 (0x0).
Press any key to close this window . . .

Output
Show output from:
Build succeeded
1>----- Build s
1>Skipping analy
1>labassign -> C
===== Build
===== Build

Build succeeded
```

Q 4. Create an abstract class Employee with abstract method CalculateSalary().

- Derive classes FullTimeEmployee and PartTimeEmployee and implement the salary calculation.
- Instantiate objects and calculate salary for both types.

Ans 4. using System;

namespace LabAssignment2

{

abstract class Employee

{

public string Name { get; set; }

public int ID { get; set; }

public Employee(string name, int id)

{

 Name = name;

 ID = id;

}

public abstract double CalculateSalary();

public void DisplayInfo()

{

 Console.WriteLine(\$"Employee Name: {Name}");

 Console.WriteLine(\$"Employee ID: {ID}");

}

}

class FullTimeEmployee : Employee

```

{
    public double MonthlySalary { get; set; }

    public FullTimeEmployee(string name, int id, double monthlySalary)
        : base(name, id)
    {
        MonthlySalary = monthlySalary;
    }

    public override double CalculateSalary()
    {
        return MonthlySalary;
    }
}

class flartTimeEmployee : Employee
{
    public double HourlyRate { get; set; }
    public int HoursWorked { get; set; }

    public flartTimeEmployee(string name, int id, double hourlyRate, int
hoursWorked)
        : base(name, id)
    {
        HourlyRate = hourlyRate;
        HoursWorked = hoursWorked;
    }

    public override double CalculateSalary()
    {
        return HourlyRate * HoursWorked;
    }
}

class flrogram
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== EMfILOYEE SALARY SYSTEM ===\n");

        FullTimeEmployee emp1 = new FullTimeEmployee("Alice Johnson", 101,
5000);

        flartTimeEmployee emp2 = new flartTimeEmployee("Bob Smith", 102, 25,
80);

        Console.WriteLine("Full-Time Employee:");
        emp1.DisplayInfo();
        Console.WriteLine($"Monthly Salary: ${emp1.CalculateSalary():0.00}");
        Console.WriteLine(new string('-', 30));

        Console.WriteLine("flart-Time Employee:");
        emp2.DisplayInfo();
        Console.WriteLine($"Calculated Salary:
${emp2.CalculateSalary():0.00}");
        Console.WriteLine(new string('-', 30));

        Console.WriteLine("flrogram finished successfully!");
        Console.WriteLine("Developed by Manish Parmar.");
    }
}

```



```

}
}
}

Microsoft Visual Studio Debug Console
labassign

Pr === Employee Salary Calculation ===
Full-Time Employee: Manish Parmar (ID: 101)
Monthly Salary: ₹50000

Part-Time Employee: Aarti Sharma (ID: 102)
Weekly Salary: ₹20000

Developed by Manish Parmar

C:\Users\Dell\source\repos\labassign\labassign\bin\Debug\net8.0\labassign.exe (process 15524) exited with code 0 (0x0).
Press any key to close this window . . .

Build succeeded
27°C Sunny
12:04 PM 10/24/2025

```

Output:

Q 5. You are building a student record system.

- Create a class Student with overloaded constructors:
 - Default constructor
 - Constructor with name and roll number
 - Constructor with name, roll number, and marks
- Display all student data using a method.

Ans 5. using System;

namespace LabAssignment2

```

{
    class Student
    {

        public string Name { get; set; }
        public int RollNumber { get; set; }
        public double Marks { get; set; }

        public Student()
        {
            Name = "Unknown";
            RollNumber = 0;
            Marks = 0.0;
        }

        public Student(string name, int rollNumber)

```

```

    {
        Name = name;
        RollNumber = rollNumber;
        Marks = 0.0;
    }

    public Student(string name, int rollNumber, double marks)
    {
        Name = name;
        RollNumber = rollNumber;
        Marks = marks;
    }

    public void DisplayData()
    {
        Console.WriteLine($"Name: {Name}");
        Console.WriteLine($"Roll Number: {RollNumber}");
        Console.WriteLine($"Marks: {Marks}");
        Console.WriteLine(new string('-', 30));
    }
}

class frogram
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== STUDENT RECORD SYSTEM ===\n");

        Student student1 = new Student();

        Student student2 = new Student("Alice Johnson", 101);

        Student student3 = new Student("Bob Smith", 102, 88.5);

        Console.WriteLine("Student 1 (Default Constructor):");
        student1.DisplayData();

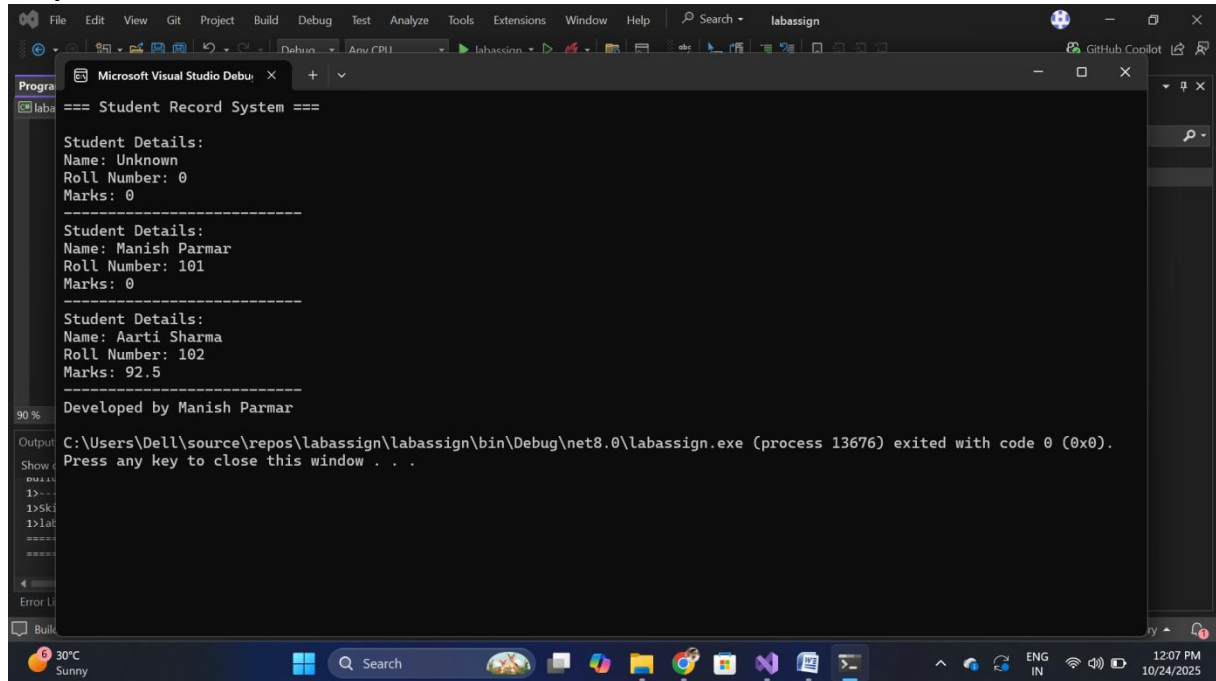
        Console.WriteLine("Student 2 (Name fi Roll Number Constructor):");
        student2.DisplayData();

        Console.WriteLine("Student 3 (Full Constructor):");
        student3.DisplayData();

        Console.WriteLine("frogram finished successfully!");
        Console.WriteLine("Developed by Manish Parmar.");
    }
}

```

Output:



The screenshot shows the Microsoft Visual Studio Debug Console. The program output is as follows:

```
=== Student Record System ===  
  
Student Details:  
Name: Unknown  
Roll Number: 0  
Marks: 0  
-----  
Student Details:  
Name: Manish Parmar  
Roll Number: 101  
Marks: 0  
-----  
Student Details:  
Name: Aarti Sharma  
Roll Number: 102  
Marks: 92.5  
-----  
Developed by Manish Parmar
```

The Output window at the bottom shows the command prompt output:

```
C:\Users\Dell\source\repos\labassign\labassign\bin\Debug\net8.0\labassign.exe (process 13676) exited with code 0 (0x0).  
Press any key to close this window . . .
```

Q 6. Develop a class Product for an inventory system.

- Include auto-implemented properties: ProductID, ProductName, Price, and Quantity.
- Use property validation logic (e.g., Price cannot be negative).
- Write a method to print product details and test it with valid and invalid inputs.

Ans 6. using System;

namespace LabAssignment2

{

class flproduct
{

private int productID;
private string productName;
private double price;
private int quantity;

public int flproductID

{

get { return productID; }

set

{

if (value <= 0)

zero.");

else

productID = value;

}

}

public string flproductName

```

    {
        get { return productName; }
        set
        {
            if (string.IsNullOrEmpty(value))
                Console.WriteLine("✗ flroduct Name cannot be empty.");
            else
                productName = value;
        }
    }

    public double flrice
    {
        get { return price; }
        set
        {
            if (value < 0)
                Console.WriteLine("✗ flrice cannot be negative.");
            else
                price = value;
        }
    }

    public int Quantity
    {
        get { return quantity; }
        set
        {
            if (value < 0)
                Console.WriteLine("✗ Quantity cannot be negative.");
            else
                quantity = value;
        }
    }

    public void Displayflroduct()
    {
        Console.WriteLine("\n=== flRODUCT DETAILS ===");
        Console.WriteLine($"flroduct ID : {flroductID}");
        Console.WriteLine($"flroduct Name : {flroductName}");
        Console.WriteLine($"flrice      : ${flrice}");
        Console.WriteLine($"Quantity  : {Quantity}");
        Console.WriteLine(new string('-', 30));
    }
}

class flrogram
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== INVENTORY SYSTEM ===\n");

        flroduct product1 = new flroduct
        {
            flroductID = 101,
            flroductName = "Laptop",
            flrice = 1200.50,
            Quantity = 10
        };
    }
}

```

```

        flproduct product2 = new flproduct
        {
            flproductID = -5,
            flproductName = "",
            flprice = -50,
            Quantity = -3
        };

        Console.WriteLine("flproduct 1 (Valid):");
        product1.Displayflproduct();

        Console.WriteLine("flproduct 2 (Invalid):");
        product2.Displayflproduct();

        Console.WriteLine("flrogram finished successfully!");
        Console.WriteLine("Developed by Manish Parmar.");
    }
}
}
}

```

Output:

```

Microsoft Visual Studio Debug Console
labassign

77  === Product Inventory System ===
78
79
80
81  === Product Details ===
82  Product ID   : 101
83  Product Name : Laptop
84  Price       : 755000
85  Quantity    : 10
86
87  -----
88  ? Invalid Product ID! It must be greater than 0.
89  ? Product name cannot be empty!
90  ? Invalid Price! It cannot be negative.
91  ? Invalid Quantity! It cannot be negative.
92
93  -----
94  === Product Details ===
95  Product ID   : 0
96  Product Name :
97  Price       : 0
98  Quantity    : 0
99
100 -----
101
102 Developed by Manish Parmar
103
104 C:\Users\Dell\source\repos\labassign\labassign\bin\Debug\net8.0\labassign.exe (process 19348) exited with code 0 (0x0).
105 Press any key to close this window . . .

```

Q 7. Design a basic Library Management System using OOP concepts.

- Classes: Book, Member, Library
- Implement:
 - Book lending feature
 - Member registration
 - Track available books
- Use concepts like inheritance, encapsulation, and method overriding wherever suitable.

Ans 7. `using System;`
`using System.Collections.Generic;`

`namespace LabAssignment2`

```
{  
    class Book  
    {  
        public int BookID { get; private set; }  
        public string Title { get; private set; }  
        public string Author { get; private set; }  
        public bool IsAvailable { get; private set; }  
  
        public Book(int bookID, string title, string author)  
        {  
            BookID = bookID;  
            Title = title;  
            Author = author;  
            IsAvailable = true;  
        }  
  
        public void Borrow()  
        {  
            if (IsAvailable)  
            {  
                IsAvailable = false;  
                Console.WriteLine($"✔ '{Title}' has been borrowed.");  
            }  
            else  
            {  
                Console.WriteLine($"✘ '{Title}' is currently not available.");  
            }  
        }  
  
        public void ReturnBook()  
        {  
            IsAvailable = true;  
            Console.WriteLine($"📞 '{Title}' has been returned and is now  
available.");  
        }  
  
        public virtual void DisplayInfo()  
        {  
            Console.WriteLine($"[{BookID}] {Title} by {Author} - {(IsAvailable o  
"Available" : "Borrowed")}");  
        }  
    }  
  
    class Member  
    {  
        public int MemberID { get; private set; }  
        public string Name { get; private set; }  
        public List<Book> BorrowedBooks { get; private set; }  
  
        public Member(int memberID, string name)  
        {  
            MemberID = memberID;  
            Name = name;  
            BorrowedBooks = new List<Book>();  
        }  
    }  
}
```

```

public void BorrowBook(Book book)
{
    if (book == null)
    {
        Console.WriteLine("✗ Invalid book.");
        return;
    }

    if (book.IsAvailable)
    {
        book.Borrow();
        BorrowedBooks.Add(book);
    }
    else
    {
        Console.WriteLine($"✗ '{book.Title}' is not available for
borrowing.");
    }
}

public void ReturnBook(Book book)
{
    if (BorrowedBooks.Contains(book))
    {
        book.ReturnBook();
        BorrowedBooks.Remove(book);
    }
    else
    {
        Console.WriteLine($"⚠ You haven't borrowed '{book.Title}'.");
    }
}

public void DisplayBorrowedBooks()
{
    Console.WriteLine($"\\nBorrowed Books by {Name}:");
    if (BorrowedBooks.Count == 0)
        Console.WriteLine("No books borrowed.");
    else
        foreach (var book in BorrowedBooks)
            Console.WriteLine($"- {book.Title}");
}

}

class Library
{
    private List<Book> books;
    private List<Member> members;

    public Library()
    {
        books = new List<Book>();
        members = new List<Member>();
    }

    public void RegisterMember(Member member)
    {
        if (member == null)
        {

```

```

        Console.WriteLine("❌ Invalid member.");
        return;
    }

    members.Add(member);
    Console.WriteLine($"👤 Member '{member.Name}' registered successfully!");
}

public void AddBook(Book book)
{
    if (book == null)
    {
        Console.WriteLine("❌ Invalid book.");
        return;
    }

    books.Add(book);
    Console.WriteLine($"📖 Book '{book.Title}' added to the library.");
}

public void DisplayAllBooks()
{
    Console.WriteLine("\n=== LIBRARY BOOKS ===");
    foreach (var book in books)
    {
        book.DisplayInfo();
    }
}

public void DisplayAllMembers()
{
    Console.WriteLine("\n=== REGISTERED MEMBERS ===");
    foreach (var member in members)
    {
        Console.WriteLine($"[{member.MemberID}] {member.Name}");
    }
}
}

class flrogram
{
    static void Main(string[] args)
    {
        Console.WriteLine("=== LIBRARY MANAGEMENT SYSTEM ===\n");

        Library library = new Library();

        Book book1 = new Book(1, "The Great Gatsby", "F. Scott Fitzgerald");
        Book book2 = new Book(2, "To Kill a Mockingbird", "Harper Lee");
        Book book3 = new Book(3, "1984", "George Orwell");

        library.AddBook(book1);
        library.AddBook(book2);
        library.AddBook(book3);
    }
}

```



```

Member member1 = new Member(101, "Alice");
Member member2 = new Member(102, "Bob");

library.RegisterMember(member1);
library.RegisterMember(member2);

library.DisplayAllBooks();
library.DisplayAllMembers();

Console.WriteLine("\n--- BOOK LENDING DEMO ---");
member1.BorrowBook(book1);
member2.BorrowBook(book1);
member1.DisplayBorrowedBooks();
member2.DisplayBorrowedBooks();

Console.WriteLine("\n--- BOOK RETURN DEMO ---");
member1.ReturnBook(book1);
member2.BorrowBook(book1);

library.DisplayAllBooks();

Console.WriteLine("\nProgram finished successfully!");
Console.WriteLine("Developed by Manish Parmar.");
}
}
}
}
}

```

Output:

```

Microsoft Visual Studio Debug Console
labassign

? Book 'C# Fundamentals' added to library.
? Book 'Python Crash Course' added to library.
? Book 'Data Structures Reference' added to library.
? Member 'Manish Parmar' registered successfully.
? Member 'Aarti Sharma' registered successfully.

=== Available Books ===
Book ID: 1, Title: C# Fundamentals, Author: Mark J. Price, Available: True
Book ID: 2, Title: Python Crash Course, Author: Eric Matthes, Available: True
[Reference] Book ID: 3, Title: Data Structures Reference, Author: Robert Lafore, Subject: Computer Science, Available: True
?? Book 'Python Crash Course' has been lent to Manish Parmar.
? Book 'Python Crash Course' is currently not available!

=== Registered Members ===
Member ID: 101, Name: Manish Parmar, Books Borrowed: 1
Member ID: 102, Name: Aarti Sharma, Books Borrowed: 0

=== Available Books ===
Book ID: 1, Title: C# Fundamentals, Author: Mark J. Price, Available: True
[Reference] Book ID: 3, Title: Data Structures Reference, Author: Robert Lafore, Subject: Computer Science, Available: True

Developed by Manish Parmar

C:\Users\Dell\source\repos\labassign\labassign\bin\Debug\net8.0\labassign.exe (process 18204) exited with code 0 (0x0).
Press any key to close this window . . .

```