Set F Solution

1. **Write a program to find maximum and minimum numbers from given array.**

```
using System;


class Program
{
   static void Main()
   {
      int[] arr = { 10, 5, 8, 22, 3, 19 };
      int max = arr[0];
      int min = arr[0];
      foreach (int num in arr)
      {
         if (num > max) max = num;
         if (num < min) min = num;
      }
      Console.WriteLine("Maximum number: " + max);
      Console.WriteLine("Minimum number: " + min);
   }
}
```

2. **Write a program to calculate hour, minutes, seconds from given seconds.**
   (Example : 3661 seconds = 1 hour : 1 minutes : 1 sec)

```
using System;
class Program
{
   static void Main()
   {
      int totalSeconds = 3661;


      int hours = totalSeconds / 3600;
      int minutes = (totalSeconds % 3600) / 60;
```

```
        int seconds = totalSeconds % 60;
        Console.WriteLine($"{totalSeconds} seconds = {hours} hour(s) : {minutes} minute(s) : {seconds}
second(s)");
    }
}
```

3. **Create a class Book with attributes BookID, Title, and Author. Write a constructor to initialize these attributes and a method DisplayBookDetails() to print these details**.

```
using System;
class Book
{
    public int BookID;
    public string Title;
    public string Author;
    public Book(int bookID, string title, string author)
    {
        BookID = bookID;
        Title = title;
        Author = author;
    }
    public void DisplayBookDetails()
    {
        Console.WriteLine($"Book ID: {BookID}, Title: {Title}, Author: {Author}");
    }
}
class Program
{
    static void Main()
    {
        Book book = new Book(101, "ABC", "XYZ");
        book.DisplayBookDetails();
    }
}
```

4. **Create a class Employee with attributes EmployeeID, Name, and Salaries. Include methods InputDetails(), DisplayDetails(), CalculateAnnualSalary(), and DeterminePerformance() to input/display details, compute the annual salary, and assign performance levels based on annual salary.**
   - **Above 10,00,000: Outstanding**
   - **5,00,000 to 10,00,000: Excellent**
   - **2,50,000 to 4,99,999: Good**
   - **Below 2,50,000: Needs Improvement**

```
using System;
class Employee{
  public int EmployeeID;
  public string Name;
  public double Salary;

  public void InputDetails() {
    Console.Write("Enter Employee ID: ");
    EmployeeID = int.Parse(Console.ReadLine());
    Console.Write("Enter Employee Name: ");
    Name = Console.ReadLine();
    Console.Write("Enter salary");
    Salary= double.Parse(Console.ReadLine());
  }

  public void DisplayDetails(){
    Console.WriteLine($"Employee ID: {EmployeeID}, Name: {Name}");
  }

  public double CalculateAnnualSalary(){
    double totalSalary = Salary*12;
    return totalSalary;
  }

  public void DeterminePerformance(){
    double annualSalary = CalculateAnnualSalary();
```

```
        Console.WriteLine("Annual Salary: " + annualSalary);

        if (annualSalary > 1000000)
            Console.WriteLine("Performance: Outstanding");
        else if (annualSalary >= 500000)
            Console.WriteLine("Performance: Excellent");
        else if (annualSalary >= 250000)
            Console.WriteLine("Performance: Good");
        else
            Console.WriteLine("Performance: Needs Improvement");
    }
}

class Program{
    static void Main(){
        Employee employee = new Employee();
        employee.InputDetails();
        employee.DisplayDetails();
        employee.DeterminePerformance();
    }
}
```

5. **Create a class ShoppingCart with methods to add items to the cart in multiple ways:**
   - **AddItem(itemName: string, quantity: int) - Adds an item by its name and quantity.**
   - **AddItem(itemName: string, quantity: int, price: double) - Adds an item by its name, quantity, and price.**
   - **AddItem(itemCode: int, quantity: int) - Adds an item by its code and quantity.**

```
using System;
using System.Collections.Generic;

class ShoppingCart
{
    private List<string> cart = new List<string>();
```

```
    public void AddItem(string itemName, int quantity)
    {
        cart.Add($"Item: {itemName}, Quantity: {quantity}");
    }
    public void AddItem(string itemName, int quantity, double price)
    {
        cart.Add($"Item: {itemName}, Quantity: {quantity}, Price: {price}");
    }
    public void AddItem(int itemCode, int quantity)
    {
        cart.Add($"Item Code: {itemCode}, Quantity: {quantity}");
    }

    public void DisplayCart()
    {
        foreach (var item in cart)
        {
            Console.WriteLine(item);
        }
    }
}
class Program
{
    static void Main()
    {
        ShoppingCart cart = new ShoppingCart();
        cart.AddItem("Laptop", 1, 800.50);
        cart.AddItem("Phone", 2);
        cart.AddItem(101, 5);
        cart.DisplayCart();
    }
}
```

6. **Create an abstract class Shape with attributes Name and Dimensions, and an abstract method CalculateArea(). Derive a class Circle that implements CalculateArea() using the formula for the area of a circle. Additionally, derive a class Rectangle that implements CalculateArea() using the formula for the area of a rectangle.**

```
using System;
abstract class Shape{
   public string Name;
   public string Dimensions;
   public abstract double CalculateArea();
}
class Circle : Shape{
   public double Radius;
   public Circle(double radius)
   {
      Name = "Circle";
      Radius = radius;
      Dimensions = "Radius: " + radius;
   }
   public override double CalculateArea()
   {
      return Math.PI * Radius * Radius;
   }
}
class Rectangle : Shape{
   public double Length;
   public double Width;
   public Rectangle(double length, double width)
   {
      Name = "Rectangle";
      Length = length;
      Width = width;
      Dimensions = "Length: " + length + ", Width: " + width;
   }
   public override double CalculateArea()
```

```
    {
        return Length * Width;
    }
}
class Program{
    static void Main(){
        Shape circle = new Circle(5);
        Console.WriteLine($"{circle.Name} Area: {circle.CalculateArea()}");
        Shape rectangle = new Rectangle(5, 3);
        Console.WriteLine($"{rectangle.Name} Area: {rectangle.CalculateArea()}");
    }
}
```

7. Write a program to count the frequency on numbers in array.(use Collections Classes)
Example :
Input : Array of int [1,2,1,1,2,2,2]
Output: no. of 1 is 3
no. of 2 is 4

```
using System;
using System.Collections;

class Program
{
    static void Main()
    {
        int[] arr = { 1, 2, 1, 1, 2, 2, 2 };
        Hashtable freqTable = new Hashtable();

        foreach (int num in arr)
        {
            if (freqTable.ContainsKey(num))
                freqTable[num] = (int)freqTable[num] + 1;
            else
                freqTable.Add(num, 1);
```

```
        }

        foreach (DictionaryEntry entry in freqTable)
        {
            Console.WriteLine($"Number {entry.Key} appears {entry.Value} times.");
        }
    }
}
```