Exercise 1: Student Attendance & Eligibility System

```
using System;
public class Attendance {
  public int AttendedDays, TotalDays, Display;
  public double Percentage;
  public Attendance(int attendedDays, int totalDays) {
    AttendedDays = attendedDays;
    TotalDays = totalDays;
  }

  public void Calc() {
    if (TotalDays > 0) {
      Percentage = (double)AttendedDays / TotalDays * 100;
    }
    else {
      Percentage = 0;
    }
  }

  public void Convert() {
    Display = (int)Percentage;
  }

  static void Main(string[] args) {
    Console.Write("Enter attended days: ");
    int attendedDays = int.Parse(Console.ReadLine());
    Console.Write("Enter total days: ");
    int totalDays = int.Parse(Console.ReadLine());

    Attendance studentAttendance = new Attendance(attendedDays, totalDays);
    studentAttendance.Calc();
    studentAttendance.Convert();
    Console.WriteLine("Attendance Percentage: " + studentAttendance.Display + "%");
  }
}
```

Exercise 2: Online Examination Result Processing

```
using System;
class OnlineExamResult
{
  static double CalculateAverage(int[] marks)
  {
    int sum = 0;
    for (int i = 0; i < marks.Length; i++)
    {
      sum += marks[i];
```

```
        }
        return (double)sum/marks.Length;
    }
    static int Convert(double average)
    {
        return (int)average;
    }

    static void Main(string[] args)
    {
        Console.Write("Enter number of subjects: ");
        int n = int.Parse(Console.ReadLine());

        int[] marks = new int[n];

        for (int i = 0; i < n; i++)
        {
            Console.Write("Enter marks for subject " + (i + 1) + ": ");
            marks[i] = int.Parse(Console.ReadLine());
        }

        double average = CalculateAverage(marks);
        Console.WriteLine("Average Marks: " + average.ToString("F2"));

        int scholarshipAverage = Convert(average);
        Console.WriteLine("Average Used for Scholarship: " + scholarshipAverage);
    }
}
```

Exercise 3: Library Fine Calculation System

```
using System;
class LibraryFineCalculator
{
    static decimal CalculateTotalFine(decimal finePerDay, int daysOverdue)
    {
        return finePerDay*daysOverdue;
    }
    static double Convert(decimal totalFine)
    {
        return (double)totalFine;
    }
    static void Main(string[] args)
    {
        Console.Write("Enter fine per day: ");
        decimal finePerDay = decimal.Parse(Console.ReadLine());

        Console.Write("Enter number of overdue days: ");
        int daysOverdue = int.Parse(Console.ReadLine());
```

```
        decimal totalFine = CalculateTotalFine(finePerDay, daysOverdue);
        Console.WriteLine("Total Fine (Decimal): " + totalFine);

        double analyticsFine = Convert(totalFine);
        Console.WriteLine("Total Fine Logged for Analytics (Double): " + analyticsFine);
    }
}
```

Exercise 4: Banking Interest Calculation Module

```
using System;
class BankAccount
{
    static decimal Calculate(decimal balance, float interestRate)
    {
        decimal rate = (decimal)interestRate;
        return balance*rate/100;
    }
    static decimal UpdateBalance(decimal balance, decimal interest)
    {
        return balance + interest;
    }
    static void Main(string[] args)
    {
        Console.Write("Enter account balance: ");
        decimal balance = Convert.ToDecimal(Console.ReadLine());
        Console.Write("Enter interest rate (from API): ");
        float interestRate = Convert.ToSingle(Console.ReadLine());
        decimal monthlyInterest = Calculate(balance, interestRate);
        balance = UpdateBalance(balance, monthlyInterest);
        Console.WriteLine("Monthly Interest: " + monthlyInterest);
        Console.WriteLine("Updated Balance: " + balance);
    }
}
```

Exercise 5: E-Commerce Order Pricing Engine

```
using System;
class ShoppingCart
{
    static double CalculateCartTotal(int itemCount)
    {
        double total = 0;
        for (int i = 1; i <= itemCount; i++)
        {
            Console.Write("Enter price of item " + i + ": ");
            double price = Convert.ToDouble(Console.ReadLine());
            total += price;
}
```

```
        return total;
    }
    static decimal ApplyTaxAndDiscount(double cartTotal, decimal taxRate, decimal discount)
    {
        decimal totalDecimal = Convert.ToDecimal(cartTotal);
        decimal taxAmount = totalDecimal * taxRate / 100;
        decimal discountedAmount = totalDecimal - discount;
        return discountedAmount + taxAmount;
    }
    static void Main(string[] args)
    {
        Console.Write("Enter number of items: ");
        int itemCount = int.Parse(Console.ReadLine());
        double cartTotal = CalculateCartTotal(itemCount);
        Console.Write("Enter tax rate (%): ");
        decimal taxRate = Convert.ToDecimal(Console.ReadLine());
        Console.Write("Enter discount amount: ");
        decimal discount = Convert.ToDecimal(Console.ReadLine());
        decimal finalAmount = ApplyTaxAndDiscount(cartTotal, taxRate, discount);
        Console.WriteLine("Final Payable Amount: " + finalAmount);
    }
}
```

Exercise 6: Weather Monitoring & Reporting

```
using System;
class TemperatureProcessing
{
    static double ConvertToCelsius(short sensorValue)
    {
        return (double)sensorValue / 10.0;
    }
    static double CalculateAverage(double[] temperatures)
    {
        double sum = 0;
        for (int i = 0; i < temperatures.Length; i++)
        {
            sum += temperatures[i];
        }
        return sum / temperatures.Length;
    }
    static int ConvertAverageForDashboard(double average)
    {
        return (int)average;
    }
    static void Main(string[] args)
    {
        Console.Write("Enter number of temperature readings: ");
        int n = int.Parse(Console.ReadLine());
        double[] temperatures = new double[n];
```

```
        for (int i = 0; i < n; i++)
        {
            Console.Write("Enter sensor reading " + (i + 1) + ": ");
            short sensorValue = Convert.ToInt16(Console.ReadLine());
            temperatures[i] = ConvertToCelsius(sensorValue);
        }
        double dailyAverage = CalculateAverage(temperatures);
        int dashboardValue = ConvertAverageForDashboard(dailyAverage);
        Console.WriteLine("Daily Average Temperature: " + dailyAverage);
        Console.WriteLine("Dashboard Display Value: " + dashboardValue);
    }
}
```

Exercise 7: University Grading Engine

```
using System;
class GradeCalculator
{
    static byte Convert(double score)
    {
        if (score < 0) score = 0;
        if (score > 100) score = 100;
        if (score >= 90) return 10;
        if (score >= 80) return 9;
        if (score >= 70) return 8;
        if (score >= 60) return 7;
        if (score >= 50) return 6;
        if (score >= 40) return 5;
        return 0;
    }
    static void Main(string[] args)
    {
        Console.Write("Enter final score: ");
        double finalScore = Convert.ToDouble(Console.ReadLine());
        byte grade = Convert(finalScore);
        Console.WriteLine("Grade: " + grade);
    }
}
```

Exercise 8: Mobile Data Usage Tracker

```
using System;
class DataUsageTracker
{
    static double ConvertBytesToMB(long bytes)
    {
        return bytes / (1024.0 * 1024.0);
    }
    static double ConvertBytesToGB(long bytes)
```

```
    {
        return bytes / (1024.0 * 1024.0 * 1024.0);
    }
    static int RoundForMonthlySummary(double value)
    {
        return (int)Math.Round(value);
    }
    static void Main(string[] args)
    {
        Console.Write("Enter data usage in bytes: ");
        long bytesUsed = Convert.ToInt64(Console.ReadLine());
        double usageMB = ConvertBytesToMB(bytesUsed);
        double usageGB = ConvertBytesToGB(bytesUsed);
        int roundedMB = RoundForMonthlySummary(usageMB);
        int roundedGB = RoundForMonthlySummary(usageGB);
        Console.WriteLine("Usage in MB: " + usageMB);
        Console.WriteLine("Usage in GB: " + usageGB);
        Console.WriteLine("Monthly Summary (MB): " + roundedMB);
        Console.WriteLine("Monthly Summary (GB): " + roundedGB);
    }
}
```

Exercise 9: Warehouse Inventory Capacity Control

```
using System;
class InventoryCheck
{
    static bool IsOverCapacity(int itemCount, ushort maxCapacity)
    {
        return itemCount > maxCapacity;
    }
    static int ConvertCapacityForReporting(ushort maxCapacity)
    {
        return (int)maxCapacity;
    }
    static void Main(string[] args)
    {
        Console.Write("Enter current item count: ");
        int itemCount = int.Parse(Console.ReadLine());
        Console.Write("Enter maximum capacity: ");
        ushort maxCapacity = Convert.ToUInt16(Console.ReadLine());
        bool overCapacity = IsOverCapacity(itemCount, maxCapacity);
        int reportedCapacity = ConvertCapacityForReporting(maxCapacity);
        Console.WriteLine("Item Count: " + itemCount);
        Console.WriteLine("Maximum Capacity: " + reportedCapacity);
        Console.WriteLine("Over Capacity: " + overCapacity);
    }
}
```

Exercise 10: Payroll Salary Computation

```
using System;
class SalaryCalculator
{
  static decimal CalculateNetSalary(int basicSalary, double allowances, double deductions)
  {
    decimal basic = Convert.ToDecimal(basicSalary);
    decimal allow = Convert.ToDecimal(allowances);
    decimal deduct = Convert.ToDecimal(deductions);
    return basic + allow - deduct;
  }
  static void Main(string[] args)
  {
    Console.Write("Enter basic salary: ");
    int basicSalary = int.Parse(Console.ReadLine());
    Console.Write("Enter total allowances: ");
    double allowances = Convert.ToDouble(Console.ReadLine());
    Console.Write("Enter total deductions: ");
    double deductions = Convert.ToDouble(Console.ReadLine());
    decimal netSalary = CalculateNetSalary(basicSalary, allowances, deductions);
    Console.WriteLine("Net Salary: " + netSalary);
  }
}
```