# VARSHA(22BCS11553)

# EXERCISE 1

**Student Attendance & Eligibility System**

```
Console.Write("Enter total number of classes: ");
int totalClasses = int.Parse(Console.ReadLine()!);

Console.Write("Enter number of classes attended: ");
int attendedClasses = int.Parse(Console.ReadLine()!);

double attendancePercentage = ((double)attendedClasses / totalClasses) * 100;

int displayPercentage = (int)Math.Round(attendancePercentage);   // Rounded
int truncatedPercentage = (int)attendancePercentage;           // Truncated

Console.WriteLine("\n--- Attendance Report ---");
Console.WriteLine($"Raw percentage: {attendancePercentage:F2}%");
Console.WriteLine($"Rounded percentage: {displayPercentage}%");
Console.WriteLine($"Truncated percentage: {truncatedPercentage}%");

string status = displayPercentage >= 75 ? "Eligible" : "Not Eligible";
Console.WriteLine($"Status: {status}");
```

# EXERCISE 2

# Online Examination Result Processing

```
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($"Enter marks for subject {i + 1}: ");
        marks[i] = int.Parse(Console.ReadLine()!);
}
//average as double
double average = (double)marks.Sum() / n;
//average with two decimal places
Console.WriteLine($"\nAverage (2 decimal places): {average:F2}");
//for scholarship eligibility
int roundedAverage = (int)Math.Round(average);   // Rounding
int truncatedAverage = (int)average;                  // Truncation
Console.WriteLine($"Rounded average (for eligibility): {roundedAverage}");
Console.WriteLine($"Truncated average (for eligibility): {truncatedAverage}");
//Eligibility check
string status = roundedAverage >= 85 ? "Eligible" : "Not Eligible";
```

```
Console.WriteLine($"Scholarship Status: {status}");
```

# EXERCISE 3

# Library Fine Calculation System

```
decimal finePerDay = 2.50m;

Console.Write("Enter days overdue: ");

int daysOverdue = int.Parse(Console.ReadLine()!);

decimal totalFine = finePerDay * daysOverdue;

Console.WriteLine($"Total fine (decimal) for user: {totalFine:F2}");

double fineForAnalytics = (double)totalFine;

Console.WriteLine($"Total fine logged (double) for analytics: {fineForAnalytics:F2}");
```

# EXERCISE 4

# Banking Interest Calculation Module

```
decimal balance = 5000.75m;

float annualRate = 4.5f;

decimal monthlyRate = (decimal)annualRate / 12;

decimal interest = balance * monthlyRate / 100;

balance += interest;

Console.WriteLine($"Monthly interest: {interest:F2}");

Console.WriteLine($"Updated balance: {balance:F2}");
```

# EXERCISE 5

## E-Commerce Order Pricing Engine

```
double cartTotal = 299.99 + 150.50;

decimal taxRate = 7.5m;

decimal discount = 20.00m;

decimal finalTotal = (decimal)cartTotal;

finalTotal += finalTotal * taxRate / 100;

finalTotal -= discount;

Console.WriteLine($"Final payable amount: {finalTotal:F2}");
```

# EXERCISE 6

## Weather Monitoring & Reporting

```
short tempReading = 200;

double tempCelsius = tempReading * 0.1;

Console.WriteLine($"Temperature in Celsius: {tempCelsius:F2}");

double[] dailyTemps = { 25.6, 26.4, 27.1 };

double dailyAverage = 0;

foreach (var t in dailyTemps) dailyAverage += t;

dailyAverage /= dailyTemps.Length;

int displayAverage = (int)Math.Round(dailyAverage);

Console.WriteLine($"Daily average (int) for dashboard: {displayAverage}");
```

# EXERCISE 7

## University Grading Engine

```
double finalScore = 72.7;

if (finalScore < 0) finalScore = 0;

if (finalScore > 100) finalScore = 100;

byte grade = (byte)Math.Round(finalScore);

Console.WriteLine($"Final score: {finalScore:F2}");

Console.WriteLine($"Grade stored as byte: {grade}");
```

# EXERCISE 8

## Mobile Data Usage Tracker

```
long dataUsageBytes = 5_500_000_000;

double usageMB = dataUsageBytes / (1024.0 * 1024);

double usageGB = dataUsageBytes / (1024.0 * 1024 * 1024);

Console.WriteLine($"Usage: {usageMB:F2} MB, {usageGB:F2} GB");

int roundedMB = (int)Math.Round(usageMB);

int roundedGB = (int)Math.Round(usageGB);

Console.WriteLine($"Monthly summary: {roundedMB} MB, {roundedGB} GB");
```

# EXERCISE 9

## Warehouse Inventory Capacity Control

int itemCount = 60000;

ushort maxCapacity = 65535;

bool withinCapacity = itemCount <= maxCapacity;

Console.WriteLine($"Within capacity: {withinCapacity}");

ushort reportCount = itemCount > 0 && itemCount <= maxCapacity ? (ushort)itemCount : maxCapacity;

Console.WriteLine($"Reported inventory: {reportCount}");

# EXERCISE 10

## Payroll Salary Computation

int basicSalary = 50000;

double allowance = 5000.75;

double deduction = 1200.50;

decimal netSalary = basicSalary + (decimal)allowance - (decimal)deduction;

Console.WriteLine($"Net salary (decimal): {netSalary:F2}");