## Critical Thinking and Problem Solving in C# Programming

### ****Task 1: Code Exploration****

I opened the code in Visual Studio 2022 and explored the structure of the program.

#### “StudentGradingProgram.cs” opened in Visual Studio, showing the Main() method.

### ****Task 2: Problem Description****

The program is designed to calculate the average score of a student from three subjects, determine the total score, and assign a corresponding grade based on the average. It prompts the user to input marks for three subjects, processes the data, and then outputs the results to the console.

**Purpose**:  
To help users calculate academic performance through basic data input, arithmetic, and conditional logic.

### ****Task 3: Code Enhancements****

#### ****Change 1: Moved Grade Logic into a Function****

**Before:**

csharp

CopyEdit

if (average >= 90)

grade = "A";else if (average >= 80)

grade = "B";// and so on...

**After:**

csharp

CopyEdit

string GetGrade(double average)

{

if (average >= 90) return "A";

if (average >= 80) return "B";

if (average >= 70) return "C";

if (average >= 60) return "D";

return "F";

}

**Why?**  
This separates logic into reusable components, improves readability, and allows reusability if needed elsewhere in the program.

#### ****Change 2: Introduced a**** Student ****Class****

**Before:**

csharp

CopyEdit

string name;int score1, score2, score3;

**After:**

csharp

CopyEdit

class Student

{

public string Name { get; set; }

public int[] Scores { get; set; }

public double Average => Scores.Average();

public int Total => Scores.Sum();

}

**Why?**  
Using a class follows OOP principles, making the code more modular and scalable. Now we can handle multiple students easily too.

#### ****Change 3: Used a List Instead of Fixed Variables****

**Before:**

csharp

CopyEdit

int score1 = int.Parse(Console.ReadLine());int score2 = int.Parse(Console.ReadLine());int score3 = int.Parse(Console.ReadLine());

**After:**

csharp

CopyEdit

List<int> scores = new List<int>();for (int i = 1; i <= 3; i++)

{

Console.Write($"Enter score {i}: ");

scores.Add(int.Parse(Console.ReadLine()));

}

**Why?**  
This reduces repetition and makes it easier to scale up the number of subjects without changing code structure.

#### ****Change 4: Added Input Validation****

**Before:**

csharp

CopyEdit

int score = int.Parse(Console.ReadLine());

**After:**

csharp

CopyEdit

int score;while (!int.TryParse(Console.ReadLine(), out score) || score < 0 || score > 100)

{

Console.Write("Invalid input. Enter a score between 0 and 100: ");

}

**Why?**  
Prevents runtime errors and ensures that only valid marks are entered by the user.

### Final Report Summary

We took a basic console application and:

Made the code cleaner with **functions**

Applied **object-oriented principles** with a Student class

Improved data handling using **lists**

Made it more robust by adding **input validation**