C# Learning Exercises - Progressive Skill Building

Exercise 1: Simple Calculator

Skills: Variables, input/output, basic math, conditionals

Create a calculator that asks for two numbers and an operation (+, -, *, /).

```csharp
```

```
using System;

class Calculator
{
    static void Main()
    {
        Console.WriteLine("=== Simple Calculator ===");

        Console.Write("Enter first number: ");
        double num1 = double.Parse(Console.ReadLine());

        Console.Write("Enter operation (+, -, *, /): ");
        string operation = Console.ReadLine();

        Console.Write("Enter second number: ");
        double num2 = double.Parse(Console.ReadLine());

        double result = 0;
        bool validOperation = true;

        switch (operation)
        {
            case "+":
                result = num1 + num2;
                break;
            case "-":
                result = num1 - num2;
                break;
            case "*":
                result = num1 * num2;
                break;
            case "/":
                if (num2 != 0)
                    result = num1 / num2;
                else
                {
                    Console.WriteLine("Error: Cannot divide by zero!");
                    validOperation = false;
                }
                break;
            default:
                Console.WriteLine("Error: Invalid operation!");
                validOperation = false;
                break;
        }
```

```
        if (validOperation)
            Console.WriteLine($"Result: {num1} {operation} {num2} = {result}");
    }
}
```

Challenge: Add more operations like power (^) or modulus (%).

Exercise 2: Number Guessing Game

Skills: Random numbers, loops, conditionals, user input validation

csharp

```
using System;

class GuessingGame
{
    static void Main()
    {
        Console.WriteLine("=== Number Guessing Game ===");
        Console.WriteLine("I'm thinking of a number between 1 and 100!");

        Random random = new Random();
        int secretNumber = random.Next(1, 101);
        int attempts = 0;
        int maxAttempts = 7;
        bool hasWon = false;

        while (attempts < maxAttempts && !hasWon)
        {
            Console.Write($"Attempt {attempts + 1}/{maxAttempts} - Enter your guess: ");

            if (int.TryParse(Console.ReadLine(), out int guess))
            {
                attempts++;

                if (guess == secretNumber)
                {
                    Console.WriteLine($"🎉 Congratulations! You guessed it in {attempts} attempts!");
                    hasWon = true;
                }
                else if (guess < secretNumber)
                {
                    Console.WriteLine("Too low! Try a higher number.");
                }
                else
                {
                    Console.WriteLine("Too high! Try a lower number.");
                }
            }
            else
            {
                Console.WriteLine("Please enter a valid number!");
            }
        }

        if (!hasWon)
        {
            Console.WriteLine($"😔 Game over! The number was {secretNumber}");
```

```
        }

        Console.WriteLine("Thanks for playing!");
    }
}
```

Challenge: Add difficulty levels (easy: 1-50, medium: 1-100, hard: 1-1000).

Exercise 3: To-Do List Manager

Skills: Lists, methods, string manipulation, menu systems

csharp

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

class TodoManager
{
    private static List<string> todoList = new List<string>();

    static void Main()
    {
        Console.WriteLine("=== To-Do List Manager ===");

        while (true)
        {
            ShowMenu();
            string choice = Console.ReadLine();

            switch (choice)
            {
                case "1":
                    AddTask();
                    break;
                case "2":
                    ViewTasks();
                    break;
                case "3":
                    RemoveTask();
                    break;
                case "4":
                    Console.WriteLine("Goodbye!");
                    return;
                default:
                    Console.WriteLine("Invalid choice! Please try again.");
                    break;
            }

            Console.WriteLine("\nPress any key to continue...");
            Console.ReadKey();
            Console.Clear();
        }
    }

    static void ShowMenu()
    {
        Console.WriteLine("\n--- MENU ---");
        Console.WriteLine("1. Add Task");
        Console.WriteLine("2. View Tasks");
```

```
    Console.WriteLine("3. Remove Task");
    Console.WriteLine("4. Exit");
    Console.Write("Choose an option: ");
}

static void AddTask()
{
    Console.Write("Enter a new task: ");
    string task = Console.ReadLine();

    if (!string.IsNullOrWhiteSpace(task))
    {
        todoList.Add(task);
        Console.WriteLine($"✅ Task '{task}' added successfully!");
    }
    else
    {
        Console.WriteLine("❌ Task cannot be empty!");
    }
}

static void ViewTasks()
{
    Console.WriteLine("\n--- YOUR TASKS ---");

    if (todoList.Count == 0)
    {
        Console.WriteLine("No tasks yet! Add some tasks to get started.");
    }
    else
    {
        for (int i = 0; i < todoList.Count; i++)
        {
            Console.WriteLine($"{i + 1}. {todoList[i]}");
        }
    }
}

static void RemoveTask()
{
    ViewTasks();

    if (todoList.Count == 0)
        return;

    Console.Write("Enter task number to remove: ");
```

```
        if (int.TryParse(Console.ReadLine(), out int taskNumber))
        {
            if (taskNumber >= 1 && taskNumber <= todoList.Count)
            {
                string removedTask = todoList[taskNumber - 1];
                todoList.RemoveAt(taskNumber - 1);
                Console.WriteLine($"✅ Task '{removedTask}' removed successfully!");
            }
            else
            {
                Console.WriteLine("❌ Invalid task number!");
            }
        }
        else
        {
            Console.WriteLine("❌ Please enter a valid number!");
        }
    }
}
```

Challenge: Add task priorities, due dates, or save tasks to a file.

Exercise 4: Simple Password Generator

Skills: Arrays, random selection, string building, parameters

csharp

```
using System;
using System.Text;

class PasswordGenerator
{
    static void Main()
    {
        Console.WriteLine("=== Password Generator ===");

        Console.Write("Password length (8-50): ");
        int length = int.Parse(Console.ReadLine());

        if (length < 8 || length > 50)
        {
            Console.WriteLine("Length must be between 8 and 50!");
            return;
        }

        Console.Write("Include uppercase letters? (y/n): ");
        bool includeUpper = Console.ReadLine().ToLower() == "y";

        Console.Write("Include numbers? (y/n): ");
        bool includeNumbers = Console.ReadLine().ToLower() == "y";

        Console.Write("Include special characters? (y/n): ");
        bool includeSpecial = Console.ReadLine().ToLower() == "y";

        string password = GeneratePassword(length, includeUpper, includeNumbers, includeSpecial);
        Console.WriteLine($"\nGenerated Password: {password}");

        // Analyze password strength
        int strength = AnalyzePasswordStrength(password);
        Console.WriteLine($"Password Strength: {GetStrengthDescription(strength)}/5");
    }

    static string GeneratePassword(int length, bool includeUpper, bool includeNumbers, bool includeSpecial)
    {
        string lowercase = "abcdefghijklmnopqrstuvwxyz";
        string uppercase = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        string numbers = "0123456789";
        string special = "!@#$%^&*()_+-=[]{}|;:,.<>?";

        StringBuilder characterSet = new StringBuilder(lowercase);

        if (includeUpper) characterSet.Append(uppercase);
        if (includeNumbers) characterSet.Append(numbers);
```

```
    if (includeSpecial) characterSet.Append(special);

    Random random = new Random();
    StringBuilder password = new StringBuilder();

    for (int i = 0; i < length; i++)
    {
        int randomIndex = random.Next(characterSet.Length);
        password.Append(characterSet[randomIndex]);
    }

    return password.ToString();
}

static int AnalyzePasswordStrength(string password)
{
    int strength = 0;

    if (password.Length >= 8) strength++;
    if (password.Length >= 12) strength++;
    if (ContainsLowercase(password)) strength++;
    if (ContainsUppercase(password)) strength++;
    if (ContainsNumbers(password)) strength++;
    if (ContainsSpecialChars(password)) strength++;

    return Math.Min(strength, 5); // Cap at 5
}

static bool ContainsLowercase(string password)
{
    foreach (char c in password)
    {
        if (char.IsLower(c)) return true;
    }
    return false;
}

static bool ContainsUppercase(string password)
{
    foreach (char c in password)
    {
        if (char.IsUpper(c)) return true;
    }
    return false;
}

static bool ContainsNumbers(string password)
```

```
    {
        foreach (char c in password)
        {
            if (char.IsDigit(c)) return true;
        }
        return false;
    }

    static bool ContainsSpecialChars(string password)
    {
        string special = "!@#$%^&*()_+-=[]{}|;:,.<>?";
        foreach (char c in password)
        {
            if (special.Contains(c)) return true;
        }
        return false;
    }

    static string GetStrengthDescription(int strength)
    {
        return strength switch
        {
            1 => "Very Weak",
            2 => "Weak",
            3 => "Fair",
            4 => "Good",
            5 => "Strong",
            _ => "Very Weak"
        };
    }
}
```

Challenge: Add option to avoid ambiguous characters (0, O, l, 1).

Exercise 5: Basic Grade Calculator

Skills: Arrays, loops, methods, error handling, formatting

```
csharp
```

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

class GradeCalculator
{
    static void Main()
    {
        Console.WriteLine("=== Grade Calculator ===");

        List<double> grades = new List<double>();

        Console.WriteLine("Enter grades (type 'done' when finished):");

        while (true)
        {
            Console.Write($"Grade {grades.Count + 1}: ");
            string input = Console.ReadLine();

            if (input.ToLower() == "done")
            {
                if (grades.Count == 0)
                {
                    Console.WriteLine("No grades entered!");
                    return;
                }
                break;
            }

            if (double.TryParse(input, out double grade))
            {
                if (grade >= 0 && grade <= 100)
                {
                    grades.Add(grade);
                    Console.WriteLine($"✅ Grade {grade} added.");
                }
                else
                {
                    Console.WriteLine("❌ Grade must be between 0 and 100!");
                }
            }
            else
            {
                Console.WriteLine("❌ Please enter a valid number!");
            }
        }
```

```
    // Calculate statistics
    DisplayGradeStatistics(grades);
}

static void DisplayGradeStatistics(List<double> grades)
{
    Console.WriteLine("\n=== GRADE STATISTICS ===");
    Console.WriteLine($"Total Grades: {grades.Count}");

    double average = grades.Average();
    double highest = grades.Max();
    double lowest = grades.Min();

    Console.WriteLine($"Average: {average:F2}%");
    Console.WriteLine($"Highest: {highest}%");
    Console.WriteLine($"Lowest: {lowest}%");
    Console.WriteLine($"Letter Grade: {GetLetterGrade(average)}");

    // Show grade distribution
    Console.WriteLine("\n--- GRADE BREAKDOWN ---");
    Console.WriteLine($"A grades (90-100): {grades.Count(g => g >= 90)}");
    Console.WriteLine($"B grades (80-89): {grades.Count(g => g >= 80 && g < 90)}");
    Console.WriteLine($"C grades (70-79): {grades.Count(g => g >= 70 && g < 80)}");
    Console.WriteLine($"D grades (60-69): {grades.Count(g => g >= 60 && g < 70)}");
    Console.WriteLine($"F grades (0-59): {grades.Count(g => g < 60)}");

    // Show all grades
    Console.WriteLine("\n--- ALL GRADES ---");
    for (int i = 0; i < grades.Count; i++)
    {
        Console.WriteLine($"{i + 1}. {grades[i]}% ({GetLetterGrade(grades[i])})");
    }
}

static string GetLetterGrade(double percentage)
{
    return percentage switch
    {
        >= 90 => "A",
        >= 80 => "B",
        >= 70 => "C",
        >= 60 => "D",
        _ => "F"
    };
```

```
    }
}
```

Challenge: Add weighted grades (tests worth more than homework).

Next Steps

Try each exercise in order. For each one:

1. **Type it out** (don't copy-paste) - this builds muscle memory
2. **Run it** and test with different inputs
3. **Break it** - try invalid inputs to see what happens
4. **Fix any issues** you find
5. **Try the challenges** to extend your learning

Which exercise would you like to start with?