BCIT™

Object Oriented Development with .NET

DAY 1 EXERCISES

25 MARKS

Submission Details:	Please upload this document with your answers to the appropriate drop box.
Late Penalty:	**10% deducted each day this assignment is late so you can still submit late and get a decent mark within a reasonable time frame**.

GitHub repositories

I have stored the source code in my GitHub repositories for exercises 1 and 2 today.

Thanks to the Visual Studio integrated Git functionality you can easily pull the code from GitHub into Visual Studio. For instructions on how to do this, go to today's lesson on the Learning Hub and download the **Visual Studio - Cloning from GitHub** document.

Exercise 1 (3 marks)

Inside Visual Studio, clone the repository found at this location:

https://github.com/cwatson-bcit/Csharp_Day1_Exe1

There are several syntax errors in the source code. You need to correct the syntax errors before running the console application.

Executing the corrected code will produce the following output:

```
Enter Employee Details
----------------------
Enter Employee ID (3 digits) : 123
Enter Employee Name          : Craig
Enter Employee Annual Salary : $21234
Enter Employee Department    : Sales


Your Entered Employee Details
-----------------------------
Employee ID Is         : 123
Employee Name Is       : Craig
Employee Salary Is     : $21234
Employee Department Is : Sales
```

Paste the contents of your Program.cs file in the box below after you have corrected the code.

(No Screenshot please)

```
using System;
namespace ConsoleApp
{
    class Program
    {
        static void Main()
        {
            //Ask User to Enter the Employee Details
            Console.WriteLine("Enter Employee Details");
            Console.WriteLine("----------------------");
            Console.Write("Enter Employee ID (3 digits) : ");
            int employeeID = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Employee Name         : ");

            string name = Console.ReadLine();
            Console.Write("Enter Employee Annual Salary : $");
            int salary = Convert.ToInt32(Console.ReadLine());
            Console.Write("Enter Employee Department    : ");
```

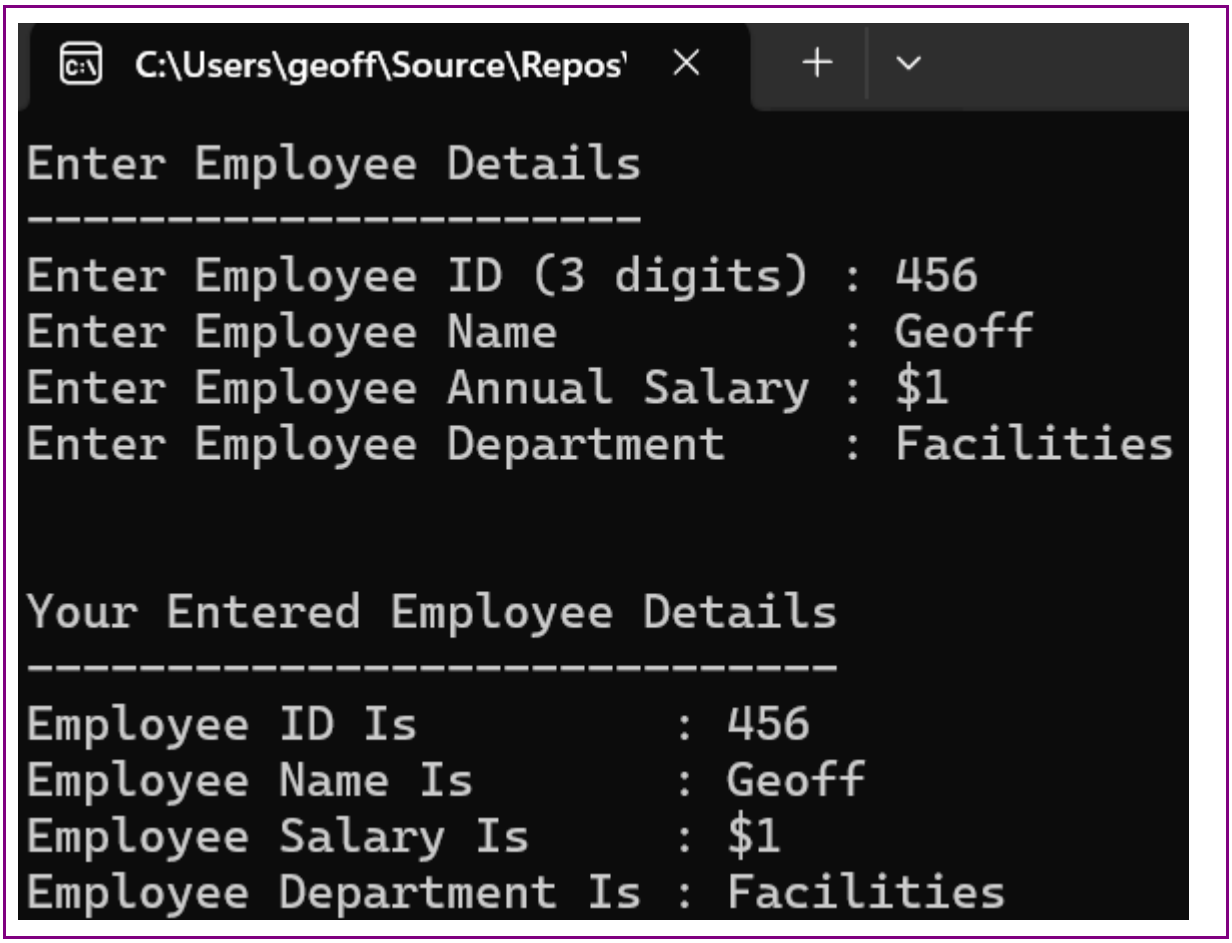
```
            string department = Console.ReadLine();

            //Display the Entered Employee Details
            Console.WriteLine(); Console.WriteLine();
            Console.WriteLine("Your Entered Employee Details");
            Console.WriteLine("-----------------------------");
            Console.WriteLine($"Employee ID Is         : {employeeID}");
            Console.WriteLine($"Employee Name Is       : {name}");
            Console.WriteLine($"Employee Salary Is     : ${salary}");
            Console.WriteLine($"Employee Department Is : {department}");
            Console.ReadLine();

        }
    }
}
```

Place a screenshot in the box below of your output after you have corrected the errors.

Make sure you enter your name as the Employee Name before taking the screen shot.


C:\Users\geoff\Source\Repos'
Enter Employee Details
Enter Employee ID (3 digits) : 456
Enter Employee Name : Geoff
Enter Employee Annual Salary : $1
Enter Employee Department : Facilities
Your Entered Employee Details
Employee ID Is : 456
Employee Name Is : Geoff
Employee Salary Is : $1
Employee Department Is : Facilities

Exercise 2 (3 marks)

Inside Visual Studio, clone my GitHub repository found at this location:

https://github.com/cwatson-bcit/Csharp_Day1_Exe2

When you execute the code, you will encounter a run time error. You need to correct the run time error then execute the console application again.

Executing the corrected code will produce the following output:

```
Enter Student Details
--------------------
Enter Name: Craig

Enter Marks of three Subjects:
Subject1: 88
Subject2: 76
Subject3: 49

Student Details are as Follows
----------------------------
Name: Craig
Total Marks : 213
Average Mark: 71
```

Paste the contents of your Program.cs file in the box below after you have corrected the code.

(No Screenshot please)

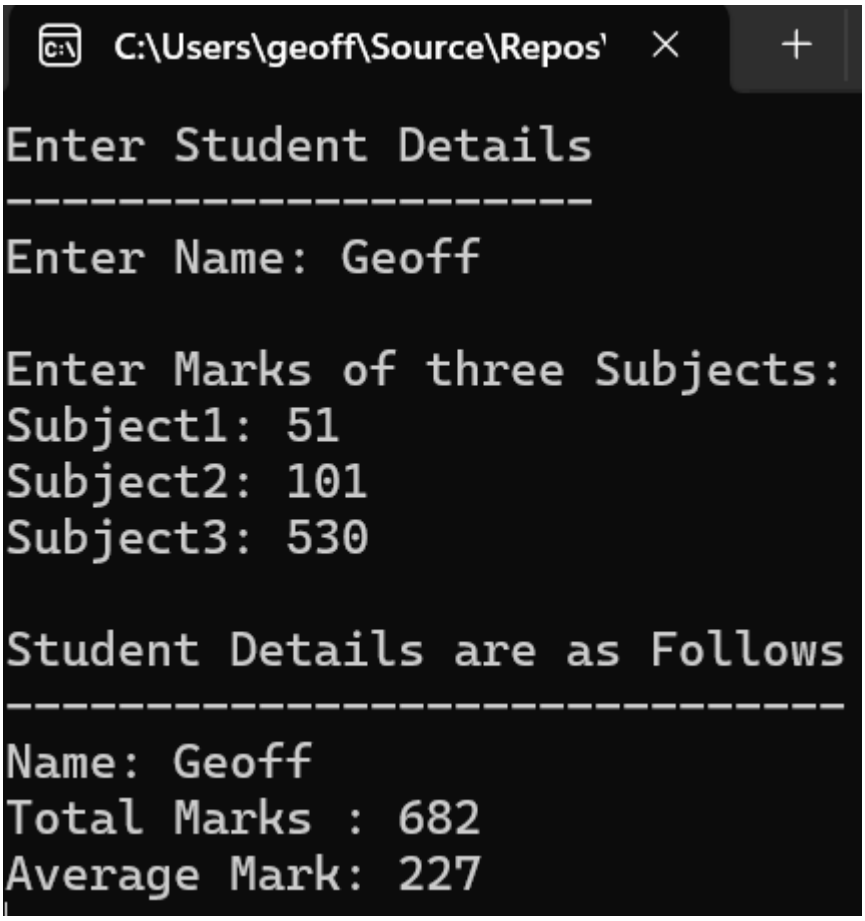
```
using System;

namespace ConsoleApp1
{
    internal class Program
    {
        static void Main()
        {
            //Ask the user to Enter Student Details
            Console.WriteLine("Enter Student Details");
            Console.WriteLine("----------------------");
            Console.Write("Enter Name: ");
            string Name = Console.ReadLine();
            Console.WriteLine("\nEnter Marks of three Subjects:");
            Console.Write("Subject1: ");
            int Mark1 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Subject2: ");
            int Mark2 = Convert.ToInt32(Console.ReadLine());
            Console.Write("Subject3: ");
            int Mark3 = Convert.ToInt32(Console.ReadLine());
            int TotalMarks = Mark1 + Mark2 + Mark3;
            int AverageMark = TotalMarks / 3;
```

```
            //Display the Student Details
            Console.WriteLine("\nStudent Details are as Follows");
            Console.WriteLine("-----------------------------");
            Console.WriteLine($"Name: {Name}");
            Console.WriteLine($"Total Marks : {TotalMarks}");
            Console.WriteLine($"Average Mark: {AverageMark}");
            Console.ReadLine();
        }
    }
}
```

Place a screenshot in the box below of your output after you have corrected the runtime error.

Make sure you enter your name as the Employee Name before taking the screen shot.



Exercise 3 (4 marks)

This exercise will help you practice variable declaration and assignment while adhering to C# coding conventions.

Develop a console application that constructs a weather forecast message by using stored values in variables.

Variable Declarations

- Declare a variable to store the name of the city ("Vancouver").
- Declare a variable to describe the weather condition ("rainy").
- Declare a variable to store the top temperature (16).

Constructing the Message

Use the declared variables to form the following message and write it to the console.

```
The weather in Vancouver is rainy with a top temperature of 16°C.
```

Guidelines

- Ensure variable names are meaningful, descriptive, and adhere to camelCase convention.
- Choose the appropriate data types for each variable:
- Ensure your code is well-indented to improve readability.
- All unnecessary code is removed.

Paste your Main() method in the box below:

(No Screenshot please)

Place a screenshot in the box below of your output.

Exercise 4 (5 marks)

This exercise will help reinforce the concepts of variable declaration, assignment, string concatenation, and console input-output.

Develop a console application that accepts a user's first name and outputs a personalized compliment.

Variable Declarations

- Declare a variable to store the user's first name.
- Declare a constant variable to hold a compliment, for instance: "is awesome!".
- Declare a third variable that will be used to store the concatenated message.

Console Input

- Prompt the user with: "Please enter your preferred name: ".
- Store the input in the first-name variable.
- Combine the first name and the compliment variables and store it in the third variable.

Console Output

Display the concatenated message on the console.

Debugging

- Set a breakpoint on your final ReadLine() statement.
- Run your application in debug mode. When the breakpoint is hit, capture a screenshot of the Locals Window.

Hints

- Variable names should be meaningful and descriptive, adhering to naming conventions.
- Ensure proper indentation in your code for readability.
- Remember that string concatenation can be achieved using the `+` operator or by using the $"{variable1} {variable2}" format.

Place a screenshot in the box below of your locals window while paused at the breakpoint. Be sure to capture the variable names, values and types.

Paste your Main method in the box below:

(No Screenshot please)

Place a screenshot in the box below of your output.

Exercise 5 (5 marks)

This exercise will reinforce your ability to work with int, double, string, and decimal types, as well as your understanding of type casting.

You will develop a console application that demonstrates the following:

- Variable declaration.
- Variable assignment
- Data type conversion.
- Arithmetic operations using different data types.

Variable Declarations

- Declare an int variable named wholeNumber and assign it a value of 10.
- Declare a double variable named doubleNumber and assign it a value of 3.14.
- Declare a string variable named numberString and assign it a value of "7.5".
- Declare a decimal variable named convertedDecimal.
    - To Convert the value in numberString to a decimal type using the Decimal.Parse() method.
- Calculate the sum of wholeNumber , doubleNumber, and convertedDecimal.
- Use the following code for your calculation:

```
decimal totalSum = wholeNumber +
                   (decimal)doubleNumber + convertedDecimal;
```

Console Output

Display the value of totalSum to the console using string interpolation. Your output should be in the following format:

```
The total sum is: 20.64
```

Hints

- Test your code with different values for numberString to ensure accurate conversion and calculations.
- Pay attention to indentation and other C# coding conventions to maintain code readability.

Paste your Main method in the box below:

(No Screenshot please)

Place a screenshot in the box below of your output.

Exercise 6 (5 marks)

This exercise will help you understand how to handle file operations for logging exceptions and other information.

Develop a console application that utilizes the StreamWriter class from the System.IO namespace to create and write log data to a file.

Setup

Add using System.IO; at the top of your Program.cs file to include the necessary namespace.

Declare Constants

- Declare a constant string variable for the file name log.txt.
- Declare a constant boolean variable to specify whether to append data to the file. Assign it true.
- Declare a constant string variable for the message to log "This is a log entry.".

Initialize StreamWriter

Create a StreamWriter object using the file path and the append flag.

Write Log

Use the StreamWriter object to write the log message to the file.

Resource Management

Don't forget to close the StreamWriter to release the resources.

Verification

Navigate to the directory where your application's solution is stored and verify that the `log.txt` file has been created and contains the log message.

Hints

- Consider wrapping the StreamWriter code in a try catch block to handle potential exceptions.
- Constants should be named using all upper-case characters.

Paste your Main method in the box below: (No Screenshot please)

Place a screenshot in the box below of the file directory containing your log file.